Editor 2.0 User Manual

© 2008, The YAWL Foundation

Document Control

Date	Author	Version	Change
2 Sep 2004	Sean Kneipp	0.1	Initial Draft
27 Sep 2004	Sean Kneipp	0.2	Data Perspective
17 Oct 2004	Sean Kneipp	0.3	Updated Editor Version
26 Oct 2004	Sean Kneipp	0.4	Updated Engine Integration
25 Nov 2004	Lindsay Bradford	0.5	Sync with Editor 1.1
04 Feb 2005	Lindsay Bradford	1	Finalised Release
25 Feb 2005	Lindsay Bradford	1.1-1	Sync with Editor 1.1-1
17 May 2005	Lindsay Bradford	1.2	Sync with Editor 1.2
27 Oct 2005	Lindsay Bradford	1.3	Sync with Editor 1.3
25 Feb 2006	Lindsay Bradford	1.4	Sync with Editor 1.4
19 May 2006	Lindsay Bradford	1.4.1	Sync with Editor 1.4.1
12 July 2006	Lindsay Bradford	1.4.2	Sync with Editor 1.4.2
10 Aug 2006	Lindsay Bradford	1.4.3	Sync with Editor 1.4.3
07 Dec 2006	Jessica Prestedge	1.4.5	Sync with Editor 1.4.5
01 Aug 2007	Lindsay Bradford	1.5	Sync with Editor 1.5
24 Jan 2008	Lindsay Bradford	2.0_beta	Sync with Editor 2.0_beta
15 May 2008	Marcello La Rosa	2.0_beta	Synch with Editor 2.0_beta release
9 July 2008	Michael Adams	2.0_beta	Additions for Release 2.0 beta.update1

How to Use This Manual

This manual describes the YAWL Process Editor, Release 2.0 version. New users are encouraged to read the manual sequentially; experienced users may use the table of contents to find information about the various Editor features.

Table of Contents

Document Control ii
How to Use This Manual ii
Table of Contents iii
Welcome to YAWL 1
    What is YAWL? 1
    Obtaining the Latest Version of YAWL 1
    The YAWL Foundation 1
    The YAWL Architecture 1
Getting Started 2
    What's New 2
    Launching the YAWL Editor 6
The YAWL Editor Workspace 7
    Menu Toolbar 7
    Specification Maintenance 7
    Specification Verification, Analysis Exporting & Importing 7
    Net Maintenance 8
    Edit Options 8
    Alignment Options 8
    Object Sizes 8
    Cancellation Sets 8
    Zoom Options 8
    Palette Bar 9
    Atomic Task 9
    Composite Task 9
    Multiple Atomic Task 9
    Multiple Composite Tasks 9
    Condition 9
    Marquee Selection 9
    Drag Net Window 10
    Status Bar 10
    Canvas 10
    Specification Problem Table 10
    Background task progress bar 10
    Menus Overview 10
    Specification 10
    Net 11
    Edit 11
    Elements 11
    Tools 11
    View 11
    Help 11
Creating Your First Specification 12
    Overview 12
    The Scenario 12
    Creating Your First Specification 12

Atomic Tasks....13
Task Decoration....15
Creating Splits and Joins....15
Composite Tasks....17
Multiple Atomic Tasks....19
Multiple Composite Tasks....21
Conditions....22
Changing the Starting Net....26
Changing the Appearance of Your Specification....27
Improving the Look of Flow Relations....27
Editing Objects....28
Changing Font Size....29
Changing Task Icons....29
Using Custom Icons....29
Advanced Specification Features....31
Cancellation Sets....31
Data Type Definitions....33
Net Decomposition Detail / Net Variables....35
Task Decomposition....36
Task Decomposition Detail / Task Variables....37
Adding a Variable to a Task....38
Task Parameters....39
A brief introduction to XQuery....43
Flow Detail....44
Multiple Instance Queries....45
Fast-Tracking Data Definition....46
End of Scenario....47
Connections....49
Connecting to the YAWL Engine....49
Connecting to the Resource Service....50
Connecting a Decomposition to a registered YAWL Service....50
Validating, Exporting and Importing....52
Specification Analysis....55
Automated task....56
Codelets....57
Manual task - Resource Management....59
Step 1....59
Step 2....60
Step 3....60
Step 4....61
Step 5....62
Timeout Task....63
Activation on enablement....63
Activation on starting....64
Expiry value....64
Extended Attributes....66
Illustrative Examples....68
Multiple Instance Task Example....68
Timeout Example....68
Known Issues....70

Troubleshooting.....71
Copyright Notice.....72
Acknowledgements.....72

Welcome to YAWL

What is YAWL?

Based on a rigorous analysis of existing workflow management systems and workflow languages, we have developed a new workflow language called YAWL (Yet Another Workflow Language). To identify the differences between the various languages, we have collected a fairly complete set of workflow patterns (http://www.workflowpatterns.com). Based on these patterns we have evaluated several workflow products and detected considerable differences in expressive power. Languages based on Petri nets perform better when it comes to state-based workflow patterns. However, some patterns (e.g. involving multiple instances, complex synchronizations or non-local withdrawals) are not easy to map onto (high-level) Petri nets. This inspired us to develop a new language by taking Petri nets as a starting point and adding mechanisms to allow for a more direct and intuitive support of the workflow patterns identified.

YAWL can be considered a very powerful workflow language, built upon experiences with languages supported by contemporary workflow management systems. While not a commercial language itself, it encompasses these languages; in addition, has a formal semantics. Such an approach is in contrast with e.g. WfMC's XPDL which takes commonalities between various languages as a starting point and does not have formal semantics. Its design hopefully allows YAWL to be used for the purposes of the study of expressiveness and interoperability issues.

Obtaining the Latest Version of YAWL

As new versions of the YAWL Engine and Editor are released to the public, they will be available for download from the YAWL Project website (http://sourceforge.net/projects/yawl). From here it is also possible to access the source code of the Engine and Editor for development purposes.

The YAWL Foundation

For more information and progress on the YAWL project, visit the YAWL Foundation Homepage (http://yawlfoundation.org). The YAWL Foundation is a non-profit organisation that acts as custodian of all IP related to YAWL and its support environment.

The YAWL Architecture

The interactions among the major components of the YAWL Environment are depicted in Fig. 1.

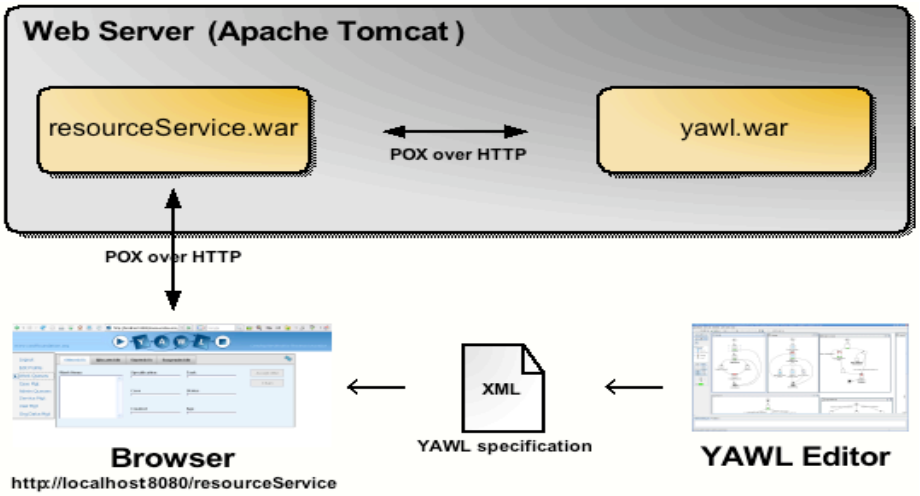


Fig. 1 The YAWL Components

Getting Started

What's New

Following is a list of new features and bug fixes introduced between this release and the previous (Version 1.5) of the YAWL Editor:

A. Change List: Release 2.0 beta → 2.0 beta.update1

- Resource Wizard dialogs now remain 'always on top' when active.
- In the Resource Dialogs, replaced all occurrences of 'Users' with 'Participants' and reworded/corrected the text where necessary.
- In the Resource Dialogs, sorted the participant and role lists. Deferred allocation list now shows net-level vars (restricted to String type only).
- In the Resource Wizard, fixed the "retain familiar" and "4-eye principle" checkboxes so they can be deselected after being selected (i.e. can now undo a previous choice).
- In Resource Wizard, added buttons to unselect all previous selections of roles and/or participants in the distribution set selection screen.
- Added the ability to mark/unmark a decomposition as 'automated' in the decomposition dialog.
- Set 'Manage Resourcing' menu to only be available when the decomposition is not 'automated'.

- Added the ability to set a 'codelet' for an automated task, which will be executed when the task is enabled.

- Modified parameter mapping dialog so that a choice can be made between mapping a task variable's value or an expression to an output parameter.

- Completed functionality of task timer setting, so that a timer can be set by fixed expiry date, duraton or deferred values via a net-level parameter. To facilitate the deferred option, a new type, YTimerType, has been included as a basic data type.

- Refactored the 'save on exit' dialog functionality so that it only appears if changes have been made since the last save.

- Removed the superfluous "Verify Password" field from "Set Resourcing Service" and "Set Engine Detail".

- Various renaming of menu items has occurred throughout.

- Modified the schema preamble in the data definition dialog by prefixing 'xs:', allowing for symbolic referencing in data schemas.

- Also in the data definition dialog, added Copy, Cut, Paste, Undo and Redo buttons and CTRL-V and CTRL-C as shortcuts; also changed the length of a TAB to 2 spaces (down from 16); and doubled the initial size of the dialog.

- Checkbox added to export specification dialog to auto increment version number. The version number is now saved when dialog exits, and is reflected in spec xml.

- The export specification dialog will now ask for a valid specification id the first time a specification is exported, if it hasn't already been set in the specification properties; 'unnamed.ywl' is no longer considered to be a valid id.

- Fixed a bug that was throwing an exception if a role name uses the "&" symbol.

- Fixed generation of export files containing simply the word 'null' if a problem occurred during xml generation. Now a warning dialog will display and the file will not be created.

- Added a 'Drop Task Decomposition' menu item to the task popup menu, which disconnects a task from a decomposition (but does not remove it from the editor session, so that it can be rejoined to the task or used by other tasks).

- Refactored code so that new connection attempts to the engine or resource service with different details reinitiates the connection (previously they were ignored). Also improved the dialog's look and feel.

- When creating a new specification, its window size will completely fit within the outer window.
- When importing a specification from xml, its window size will completely fit within the outer window.
- A warning dialog displays when a connection with the resource service is not available, and the "Manage Resourcing" menu is selected.
- Fixed a bug where, if the name of a complex type is too long, other widgets were "cramped" in the update parameters dialog.
- Added the ability to specify a 'custom form' for a task.
- Fixed a bug where open net references were being displayed in the wrong place on the View menu.
- Fixed bug where importing a specification from xml would its lose flow parameters.
- Fixed bug where importing a specification from xml with more than a single net would start non-root nets iconified.

B. Change List: Version 1.5 → Release 2.0 beta

- Upgraded the editor to use version 2.0 of the engine. This includes a complete renaming of the source package hierarchy, and breaks save-file compatibility with previous versions of the editor.
- Also included in the move to the 2.0 engine is a powerful replacement of the resource perspective. The previous YAWL environment supported 9 of the 43 identified resource patterns. This release supports 38 patterns. We have no plans to support the remaining five patterns at this time.
- Automated task added. It is now possible to differentiate between manual task – a task assigned to some human resource for its execution, and an automated task – a task which is meant to be executed by the system. Currently the automated task only supports the execution of XQuery expressions. In future we plan to extend this task to allow the execution of predefined Java codelets.
- Added a timeout feature for atomic single-instance tasks. The timeout can be activated on task's enablement or on task's starting and can be given a value in terms of absolute date or duration. The timeout works as a delay for an automated task, and forces the work item's completion for a manual task.
- Changed the specification version number to be a decimal number instead of a free-form string. This includes the export dialog now supplying the specification version as a convenience to allow a workflow de-

signer to more easily manage variations of the specifications they export.

- The palette bar can now be resized via a sliding bar, mostly to allow a clear view into a deep icon tree. However, given a couple of Swing limitations, the palette bar can no longer be detached from the base editor window.

- Via the plugin mechanism, workflow designers can now specify extra attributes (so-called *extended attributes*) to be stored on variables and decompositions which will be exported into the engine specification, and used as required for their own custom services.

- Small change to the pause mechanism used at editor startup to make better usage of system CPU.

- Fixed a bug where the path of plugin icons was absolute, stopping plugin task icons from working on machines where the plugin path is different.

- Fixed a bug where the view menu would occasionally show the wrong icon for a net.

- Fixed a bug where the paste of a copied net element would result in the copy having the same engine ID as the original, disrupting correct engine export/validation behaviour.

- Fixed a bug where engine validation was failing, claiming that max instances was smaller than min instance on multi-instance tasks when max instances was set to infinity.

- Some refactoring around saving, closing and exiting to ensure that whenever a user cancels their desire to save, close or exit, they are returned to the editor correctly.

- Fixed a bug where predicates and flow priorities were occasionally being reset on tasks with XOR/OR splits.

- The behaviour to iconify nets has been removed. This is mostly due to occasional odd behaviour witnessed with Swing that is outside the control of the programmer, causing undue grief.

- Fixed a bug where multiple-instance tasks were not being initialised properly and consequently causing a number of dialogs, and behaviours around multiple-instance tasks to fail.

- Changed the ordering of menu items on the vertex popup menu so that the "Cut" action no longer appears at the top of the menu.

- Fixed a bug where cutting a task with attached flows containing predicates, then undoing the cut, would result in a previously correctly validating specification to stop validating.

- Enhanced the editor so that it shows a broken icon if the icon specified cannot be found (useful for transferring specs between machines with different icon sets plugged in).
- The editor is now closer to (but not perfectly) rebuilding the state of the net frames on a load, as per how the workflow designer left the state on a save.
- Fixed a bug where the editor was reacting poorly to an exception being thrown in an engine client-side API call when attempting to connect to a non-existent engine.

Launching the YAWL Editor

Download the latest version from the YAWL SourceForge website:

http://sourceforge.net/projects/yawl/.

The YAWL Editor is distributed as a Java Archive (jar). Double click on the *YAWLEditor2.0.jar* file to start the application (where supported). The YAWL Editor can also be started from a command line or Terminal prompt:

java –jar YAWLEditor2.0.jar

The YAWL Editor Workspace

The first time you start the YAWL Editor, you will be presented with a blank canvas, with the instructions in the Status Bar asking you to open or create specification to begin.

Before you create your first specification, let us take a brief tour of the Editor's workspace and the elements within. The workspace is shown in Fig. 2.

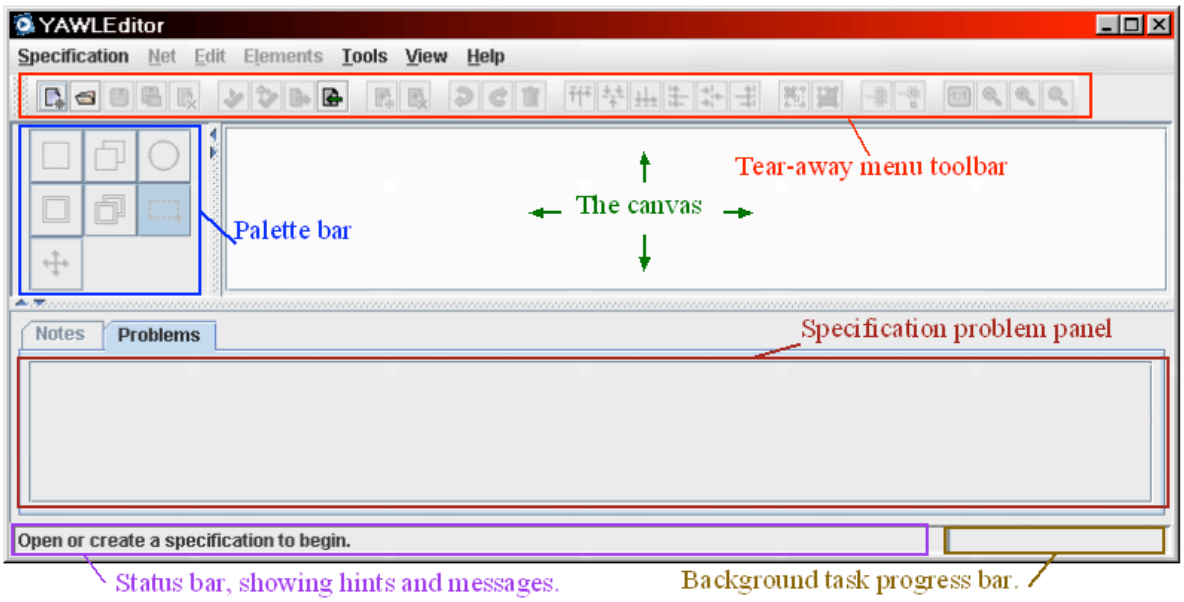


Fig. 2 The YAWL Editor Workspace

Menu Toolbar

The Menu Toolbar contains six groups of buttons to assist you in maintaining your YAWL specification. The menu can be repositioned by dragging the left-hand anchor bar.

Specification Maintenance

This group of buttons provides you the standard options to create, open, save, save-as and close YAWL specification files.

Specification Verification, Analysis Exporting & Importing

These four buttons allow you to validate your specification against YAWL syntax and semantics, analyse your specification for deadlocks and other issues, export your

workflow diagram into an XML file for uploading to the YAWL Engine, and import a YAWL Engine XML file into the editor for further alteration respectively.

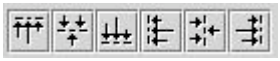
Net Maintenance

Your workflow diagrams are captured within Nets, of which a specification could contain many. You can use these buttons to create a new Net or remove existing Nets within your specification.

Edit Options

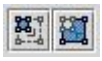
This group of buttons provides the standard Undo and Redo options as well as the option to delete the currently selected objects.

Alignment Options



These buttons can be used to assist with the alignment of objects within your specification, when multiple objects have been selected.

Object Sizes



To increase the size of an object within your specification, select the object(s) and then use these buttons.

Cancellation Sets

These buttons allow you to include in and/or exclude elements from the cancellation set of a task.

Zoom Options

These buttons allow you to apply zoom functionality to the currently selected net. In order, the buttons allow you to reset the zoom to the actual size, to zoom the entire net out, to zoom the entire net in, and to zoom into the currently selected net elements.

Palette Bar

The Palette Bar contains seven selectable buttons that assist with creation, selection and positioning of objects within your specification. This menu can be repositioned by dragging the left-hand anchor bar.

The Palette bar is also accessible by right-clicking anywhere on a net that does not contain a net element.

Once an element is selected, it is possible to drop objects in the canvas by left-clicking the mouse button.

Atomic Task

Select this button to create an Atomic Task, which represent a single task to be performed by a human or external application.

Composite Task

Select this button to create a Composite Task, which is a container for another YAWL (sub) Net - with its own set of YAWL elements constrained by the same syntax.

Multiple Atomic Task

Select this button to create a Multiple Atomic Task, which allows you to run multiple instances of a task concurrently.

Multiple Composite Tasks

Select this button to create a Multiple Composite Task, which allows you to run multiple instances of a composite task concurrently.

Condition

Select this button to create a Condition, which is a way to represent state for the Net.

Marquee Selection

Select this button to activate the Marquee Selector, which will allow you to select individual or multiple objects by clicking and dragging your left mouse button. Please note: you cannot create flows while using the Marquee Selector.

Drag Net Window

Select this button to drag the visible window of a net around that net.

Status Bar

The Status Bar provides useful hints throughout the creation of your specification and depending upon which object you are using in the Palette Bar.

Canvas

The Canvas is where you will be creating and editing your workflow diagram(s).

Specification Problem Table

The Specification problem table is where you see what problems are currently outstanding in the specification you are building.

Background task progress bar

The Background task progress bar shows work in progress for certain background tasks, like the saving of specification files.

Menus Overview

This section provides a brief overview of the YAWL Menus located along the top of the YAWL Editor.

Specification

The Specification Menu provides all the standard file options of Create, Open, Save, Save-As, Close and Exit. It also contains options to validate (engine syntax), analyse (workflow semantics), export the specification to the YAWL Engine (XML format), import specifications from the YAWL Engine XML format, update the specification's XML Schema data type definitions, set a specification's properties and print the specification.

Net

The Net Menu provides options to create, remove, set the starting net, specify decomposition detail, and resize the currently selected net. It also provides options to export a net to a PNG image file and for directly printing out the net.

Edit

The Edit Menu provides the standard options of Undo, Redo, Cut, Copy, Paste and Delete objects within your specification.

Elements

The Elements Menu allows you to align net elements within your specification, modify their size or add or remove net elements from task cancellation sets.

Tools

The Tools Menu allows you to specify a running engine that the editor should connect to so you can tailor your workflow designs to specific modules and web services installed in the engine. Included in this menu is an option for configuring specification analysis. With this release of the editor, an amount of default specification analysis is offered as part of the editor. If the **wofyawl** analysis utility is also supplied in the same directory as the editor, the configuration dialog will allow specification designers to configure and use wofyawl for additional specification analysis (Windows environments only). Also, for resource allocation, this menu offers a dialog to identify an organisation database from which resourcing detail can be retrieved.

View

You can use this menu to turn off or on the Tooltips, which provide useful hints when your mouse is positioned over a toolbar button or an option in a pop-up dialog window, obtained by right-clicking on the canvas. It also allows you to change the font size used for element labels, toggle whether diagrams should be drawn anti-aliased and toggle whether grids should be drawn with diagrams. Finally, it allows users to select one net from all available for editing.

Help

The Help Menu provides an “About the Editor” dialog, describing components used in the editor’s construction and a list of source code contributors.

Creating Your First Specification

Overview

This next chapter will lead you through the process of creating a YAWL specification from beginning to end, through a series of brief lessons following a scenario.

You can either follow all the instructions including the scenario provided, from beginning to end, or skip straight to the section that you are interested in and follow the instructions.

Look for the student icon next to the instructions for specific details of the scenario.

The Scenario

The scenario that we will be following throughout this manual is the workflow of a student who has just completed their secondary study and is now looking to start their career.

This scenario will follow the path of a student who either enrols in a University to complete their tertiary education, or undertakes private study that will eventually lead them to getting a job and starting their new career.

Creating Your First Specification

1. Click on the Create a New Specification button, , at the top left of the Menu Toolbar, or click on **Specification** in the Menu and choose **Create Specification**.

   This will bring up a blank Net called “New Net 1” which will be, by default, the starting net of the workflow. For details on selecting a starting net, consult the Changing the Starting Net section in this manual.

2. Alter the specification’s properties as you feel appropriate. Click on **Specification** in the Menu and choose **Update Specification Properties.** A screen as per Fig. 3 will appear.

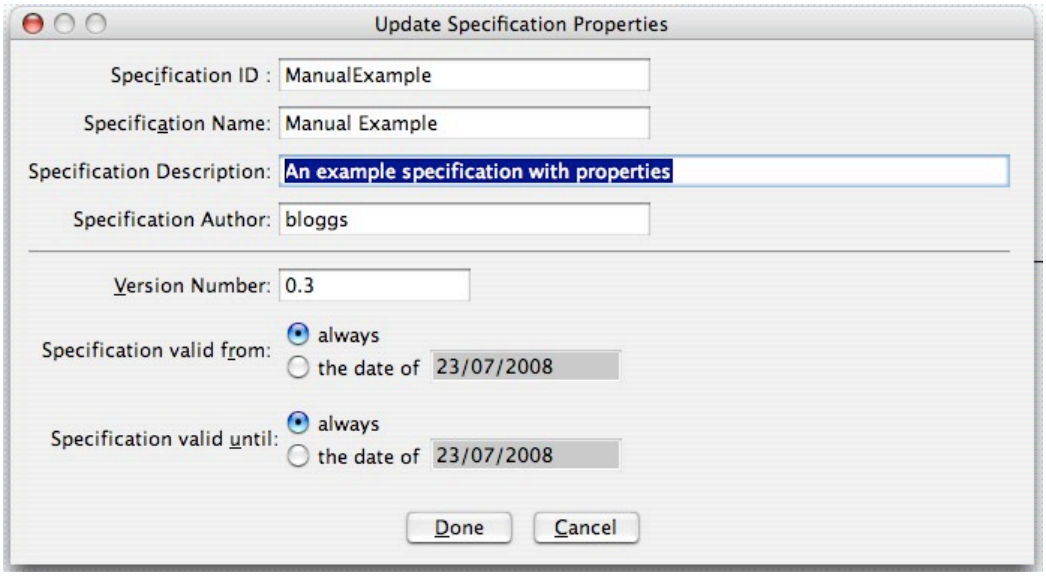


Fig. 3 Specification Properties Dialog

3. Rename this Net by clicking on the **Net** Menu and choosing **Update Net Detail**.

4. Enter the new name of the Net in the "Decomposition Label" field, then click the **Done** button.

   Decomposition Variables will be explained later in the "Net Decomposition Detail" section of this manual.

   Change the name of the Net in the Decomposition Label, to "My Career". This Net will be the primary net for our scenario.

5. You are now ready to start drawing your specification.

Atomic Tasks

1. Click on the "Add an Atomic Task" button, ☐ , in the Palette Bar, or right click in the whitespace of the Net and choose **Atomic Task**.

2. Position your mouse just to the right of the Input Condition (the ▶ symbol), and click the left mouse button once to place an Atomic Task

3. Set the decomposition of this task by right clicking on the Atomic Task and choosing **Select Task Decomposition**. You should see a dialog as per

4. **Fig. 4**.

5. Press the **Create…** button, and in the following window, "Update Task Decomposition", enter the decomposition's label.

See the "Select Task Decomposition" section of this manual for a full explanation of its features.

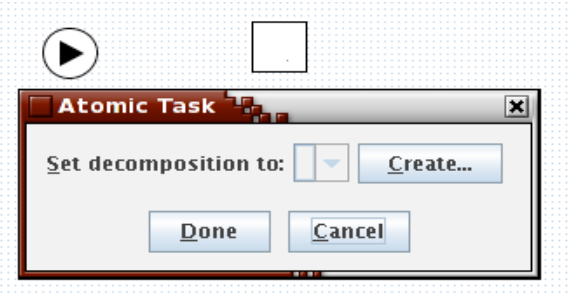


Fig. 4 The "Select Task Decomposition" dialog

6. Set the label to "Begin My Career", and click the **Done** button.

7. Note that by default, a task takes on the label of the decomposition that it is tied to (several tasks are allowed to share the same decomposition). Once you've created your task, you are free to relabel the task to whatever you like. This can be done by right-clicking on the task and choosing **Set Label…** from the pop-up menu. This will not change the name of the decomposition to which the task is tied.

8. Connect the Input Condition to your Atomic Task, as shown in Fig. 5, by finding the flow connectors that appear as small blue boxes as you hover your mouse over the sides of the objects. Hold the left mouse button down and draw a line from the flow connector on the Input Condition to the one on the Atomic Task. The editor will only show a connection point if it is valid to draw a flow connection using the current object.

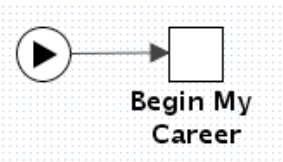


Fig. 5 An established flow relation

That's it! Your Atomic Task is set.

9. Repeat the process for the following Atomic Tasks in order: Go to University, Get A Job, Begin My Career

10. Link the Begin My Career task to the Output Condition (the ◉ symbol), as per **Fig. 6**.

11. Finally check the validity of specification by clicking on the Validate this Specification button, , in the Menu Toolbar or click on **Specification** in the Menu and choose **Validate Specification**. If all things are going

to plan, then you should receive a confirmation saying that there were no errors detected.

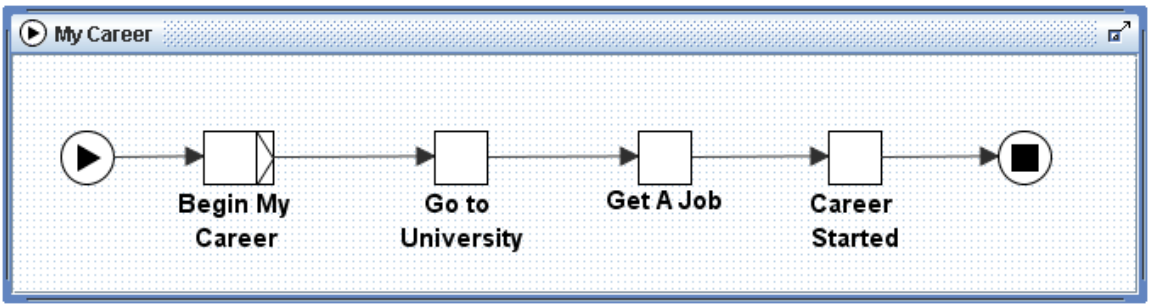


Fig. 6 The "My Career" Net

Task Decoration

Decorating a task is the process of adding splits or join conditions on task.

By putting a split on a task, you are telling the task that when it has been completed, its succeeding task could be one or more tasks. Here are the possible splits for a task:

- No split (no split on task)
- AND split
- OR split
- XOR split

By putting a join on a task, you are telling the task that it could become available through the completion of one or more preceding tasks (depending on the type of join). Below are some possible joins for a task:

- No join (no join on task)
- AND join
- OR join
- XOR join

For more information on join and split types, please consult the YAWL technical papers on the YAWL website.

Creating Splits and Joins

To create a split or join:

1. Ensure that your palette is in "Net Element Selection" mode.

2. Select a task. When a single task is selected the palette menu will expand (see Fig. 7) to include two tabs that allow you to decorate a task with a split and/or join. You can also choose a fill-colour to help visually differentiate splits from joins with the expanded palette.

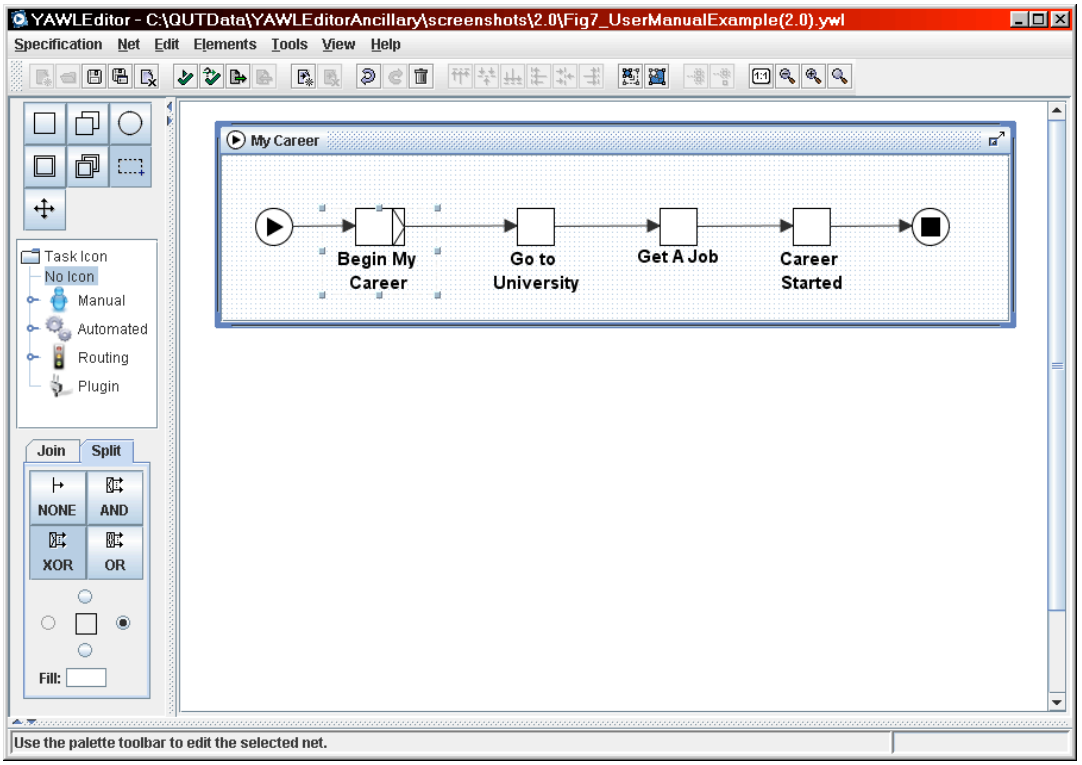


Fig. 7 Decorating a Task

1. Choose the required split or join and the orientation for the split or join to appear and then click **Done**.

2. Select your "Begin My Career" task and change the split to **Xor Split**. Then set the orientation to eastern edge of the task, as per Fig. 8.

Create a new Atomic task called "Do Private Study". This task will represent those students that choose not to go to University.

Finally, select your "Get A Job" task and decorate it with an **Xor Join**. Then set the orientation of the join to the western edge of this task.

3. Split and Join decorators allow you to connect several Flow Relations from and to your task respectively.

Create a flow relation from "Begin My Career" to "Do Private Study", then create another flow relation from "Do Private Study" to "Get A Job", as per Fig. 8.

4. Don't forget to check the validity of your specification.

Hint: If you are having trouble with positioning your tasks, the alignment tools are a big help.

When "Begin My Career" has been completed, a choice must be made on which of the two tasks ("Go To University" or "Do Private Study") will be followed (XOR Split). "Get a Job" will become available after the completion of the task selected at the point of the XOR split.

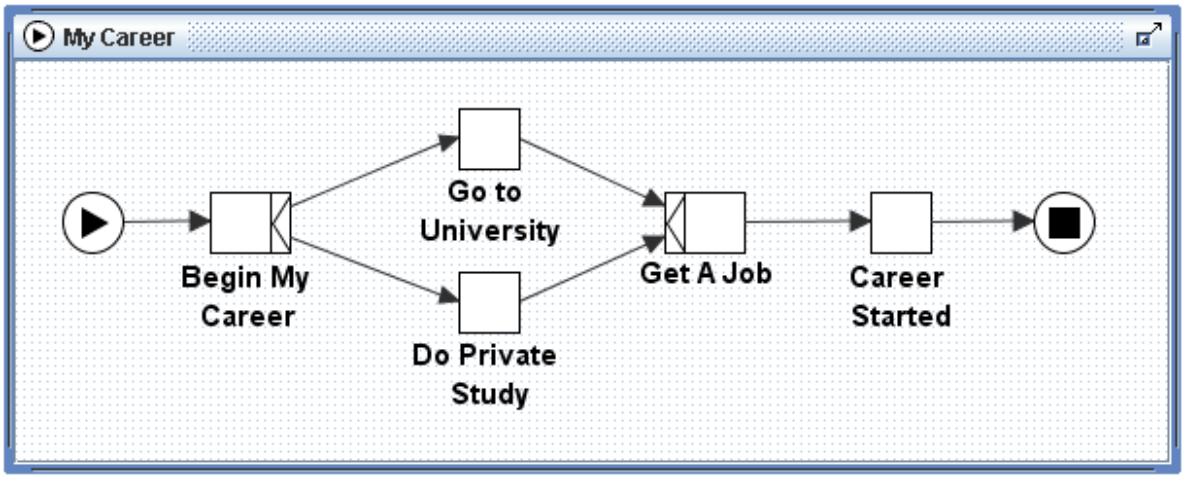


Fig. 8 XOR Split and Join

Composite Tasks

1. Composite tasks are placeholders for other YAWL Nets. That is, you can create another workflow in a separate Net, which is represented by the composite task in first net.

2. To create a Composite Task: click on the Composite Task button, , in the Palette Bar or right click in the whitespace of the Net and choose **Composite Task**.

We are going to replace our existing "Go to University" Atomic Task with a Composite Task, so choose the Enter Marquee Selection Mode button, , on the Palette Bar, click on the "Go to University Atomic Task" and press the Delete key on the keyboard. We will add in the new composite task next.

3. Place your Composite Task in your Net.

Reconnect the Flow Relations from "Begin My Career" to the new Composite Task, and from new Composite Task to "Get a Job".

4. Create a new Net by clicking on the Create a new Net button, , on the Menu Toolbar, or click on **Net** in the Menu and choose **Create Net**.

5. Choose a name for this Net by clicking on the **Net** Menu and choosing **Update Net Detail**.

We are going to call this new Net “Attend University”.

6. Return to your original Net and right click on your Composite Task and choose **Unfold to net…**. You will then be given a drop-down list with all the Nets available – choose the Net this task is to represent and then click **Done**.

Choose “Attend University”.

7. You can now fill out the detail of your new ’’Attend University’’ Net.

Create the following Atomic Tasks in order and then link them with Flow Relations and don’t forget to check for validity:

- Enrol
- Do Subjects
- Pass All Subjects
- Get Degree

The resulting nets are shown in Fig. 9.

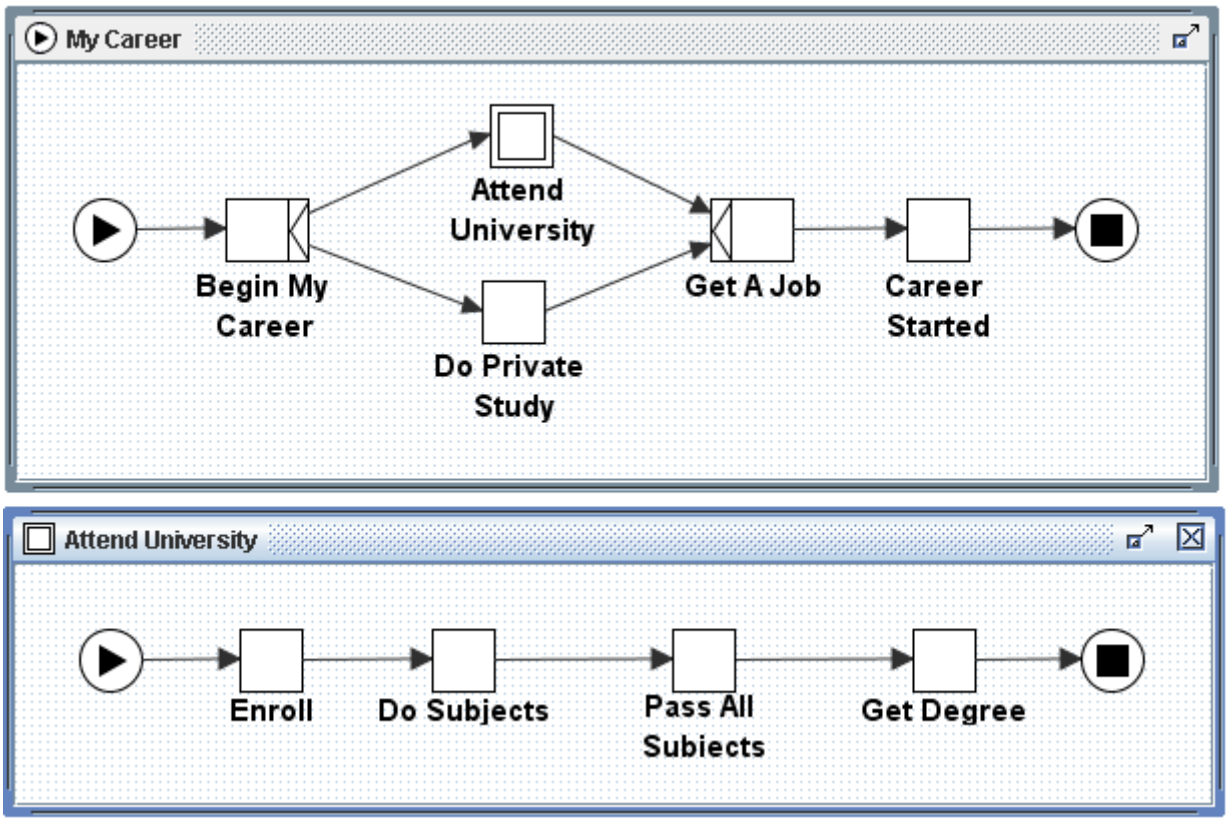


Fig. 9 "Attend University" Composite Task and its matching Net

Multiple Atomic Tasks

Multiple Atomic Tasks allow you to run multiple instances of a task concurrently.

To create a Multiple Atomic Task:

1. Click on the Add Multiple Atomic Task button, , in the Palette Bar or right click in the whitespace of the Net and choose **Multiple Atomic Task**.

Go back to the "My Career" Net. We are going to replace our existing "Do Private Study" Atomic Task, with a Multiple Atomic task, so choose the Enter Marquee Selection Mode button, , on the Palette Bar, click on the "Do Private Study" Atomic Task and press the Delete key on the keyboard. We will add in the new Multiple Atomic task next.

2. Place your Multiple Atomic Task in your Net and set the name of this task by right clicking on the task and choosing **Select Task Decomposition**.

Call this task the same name as before by selecting the "Do Private Study" decomposition from the drop-down list.

Reconnect the Flow Relations from "Begin My Career" to "Do Private Study", and from "Do Private Study" to "Get A Job", as per Fig. 10.

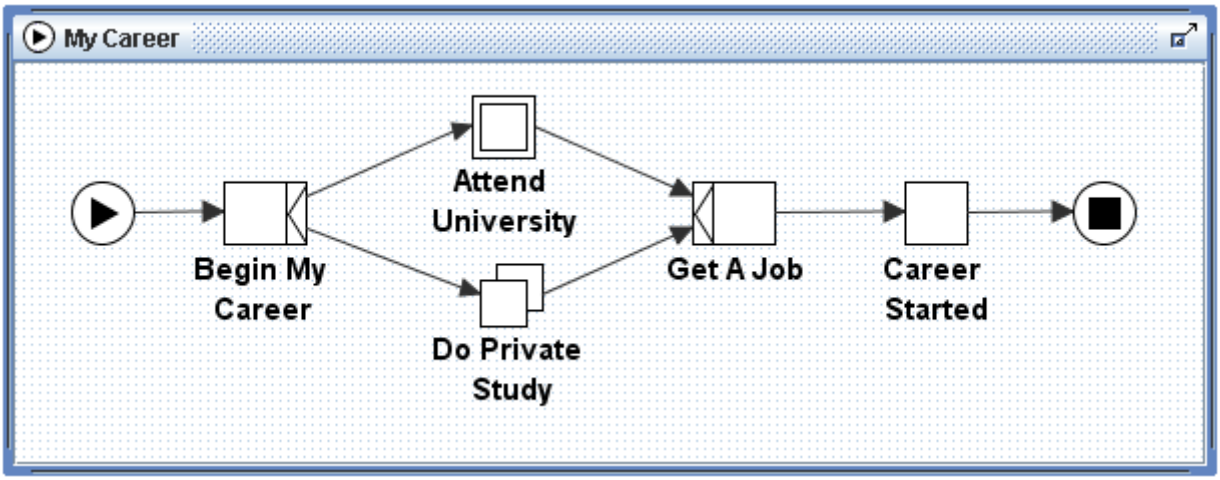


Fig. 10 Adding a Multiple Atomic Task

3. You will now need to set the parameters of the Multiple Atomic Task. Right click on the task and choose **Set Instance Detail**. Ensure that you are viewing the "Bounds" tab of the dialog, as per Fig. 11.

4. Set the Minimum Instances value. This is the minimum number of instances of this task that will be started when the task needs processing.

Set the Minimum Instances to 5.

5. Set the Maximum Instances value. This is the maximum number of instances of this task that can be created.

Set the Maximum Instances to 100.

6. Set the Continuation Threshold value. The moment all instances created have completed or as many instances as the Continuation Threshold specified have completed, the multiple instance task itself is considered completed, which triggers relevant outgoing flow relations from this task.

Set the Continuation Threshold to 50.

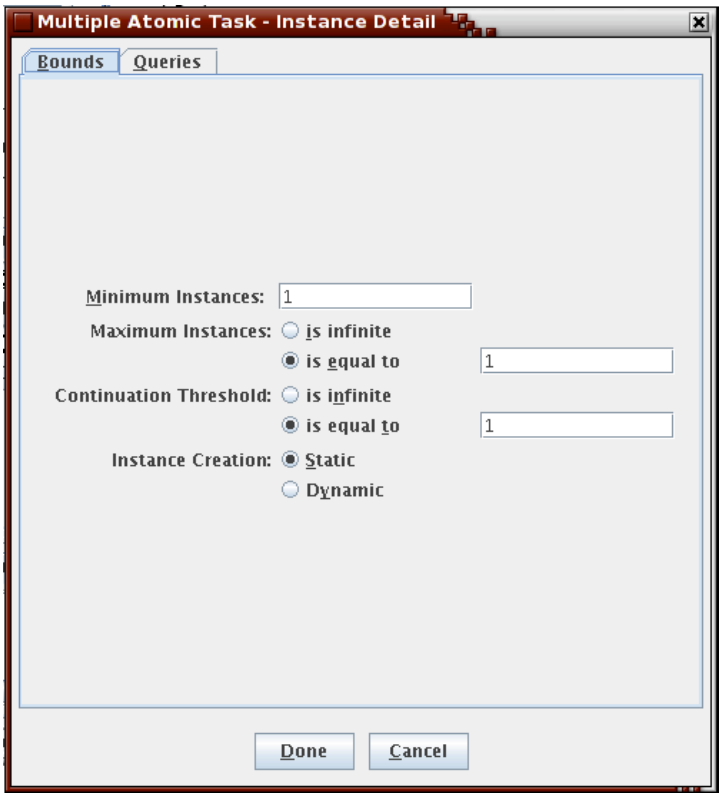


Fig. 11 Instance Bounds on Multiple Instance Tasks

7. Choose the Instance Creation. If a multi-instance task has static creation mode, when the task is started the number of instances created is decided then, and cannot be altered during the execution of the task. Conversely, if a task has dynamic creation mode then new instances of the task can start once the initial minimum instance number of tasks has started, but before the maximum number has been received.

Set the Instance Creation type to “Static”.

8. Click **Done**.

With the values set in the scenario, the Do Private Study task is indicating that a maximum of 100 instances of this task can be executed before triggering the task’s

outgoing flow. A minimum of five instances will be created, and once 50 instances have completed, the outgoing flow relations to ''Get A Job" will trigger.

The "Queries" tab of the Multiple Instance dialog will be explained later in the "Setting Multiple Instance Queries" section of this manual.

Multiple Composite Tasks

Multiple Composite Tasks allow you to run multiple instances of the Net represented by a multiple composite task, concurrently.

To create a Multiple Composite Task:

1. Click on the Add Multiple Composite Task button, , in the Palette Bar or right click in the whitespace of the Net and choose **Multiple Composite Task**.

Go to the "My Career" Net. We are going to replace our existing "Do Private Study" Multiple Instance Task, with a Multiple Composite task, so choose the Enter Marquee Selection Mode button, , on the Palette Bar, click on the "Do Private Study" Task and press the Delete key on the keyboard. We will add in the new Multiple Composite task next.

2. Place your Multiple Composite Task in your Net.

Reconnect the Flow Relations from "Begin My Career" to the new Multiple Composite Task, and from the new Multiple Composite Task to "Get a Job".

3. You will now need to set the parameters of the Multiple Composite Task. Right click on the task and choose **Set Instance Detail**.

4. Set the Minimum Instances value. This is the minimum number of instances of this task that will be started when the task needs processing.

Set the Minimum Instances to 5.

5. Set the Maximum Instances value. This is the maximum number of instances of this task that can be created.

Set the Maximum Instances to 100.

6. Set the Continuation Threshold value. The moment all instances created have completed or as many instances as the Continuation Threshold specified have completed, the multiple instance task itself is considered completed, which triggers relevant outgoing flow relations from this task.

Set the Continuation Threshold to 50.

7. Choose the Instance Creation. If a multi-instance task has static creation mode, when the task is started the number of instances created is

decided then, and cannot be altered during the execution of the task. Conversely, if a task has dynamic creation mode then new instances of the task can start once the initial minimum instance number of tasks has started, but before the maximum number has been received.

Set the Instance Creation type to “Static”.

8. Click **Done**.

9. Create a new Net by clicking on the Create a new Net button, , on the Menu Toolbar, or click on **Net** in the Menu and choose **Create Net**.

10. Give the new Net a name by clicking on the **Net** Menu and choosing **Update Net Detail.**

We are going to call this new Net “Study Privately”.

11. Return to your original Net and right click on your Multiple Composite Task and choose **Unfold to Net**. You will then be given a drop-down list with all the Nets Available – choose the Net for this task to initiate and then click **Done**.

Choose “Study Privately”:

12. You can now complete your new “Study Privately” Net represented by your Composite Task.

Create the following Atomic Tasks in order and then link them with Flow Relations as per Fig. 12:

- Read a Book
- Feel Smarter

Don’t forget to validate your specification.

Conditions

Conditions represent a states of the workflow and can be located in-between tasks.

To create a Condition:

1. Click on the Add a Condition button, , in the Palette Bar or right click in the whitespace of the Net and choose **Condition**.

Go to the “Study Privately” Net. We are going to place a loop Condition after the Read a Book Atomic Task, to determine whether we gained any knowledge from the book. We will add the new Condition next.

2. Place your Condition in your Net and set the name by right clicking on the Condition and choosing **Set Label**.

Call this Condition “Knowledge Gained?”

3. Now link to the Condition using the Flow Relations object.

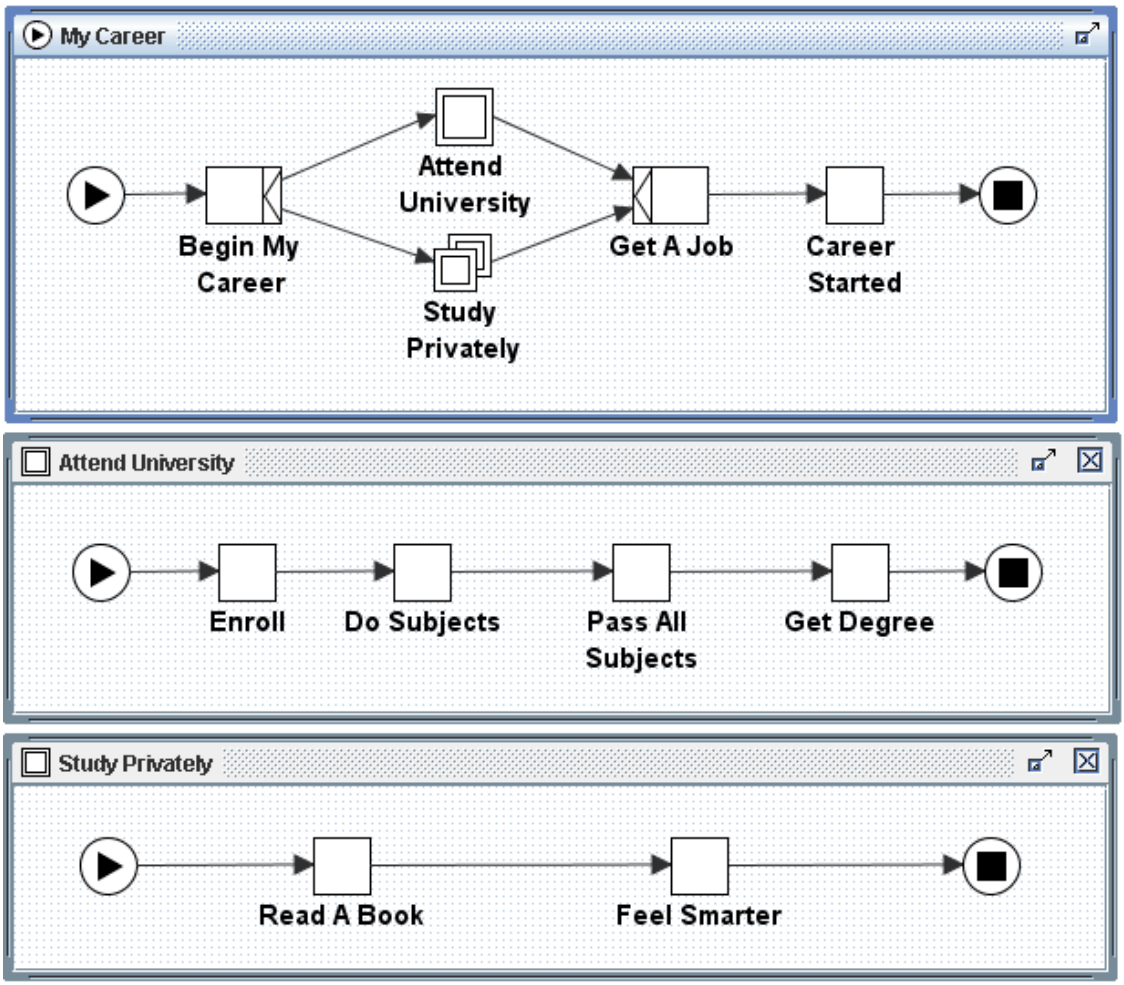


Fig. 12 "Study Privately" Multiple Composite Task and its Net

Select the Enter Marquee Selection Mode button, , from the Palette Bar and click on the Flow Relation going from the Read a Book Atomic Task to the Feel Smarter Atomic Task and press the Delete button on the keyboard.

Create a Flow Relation from “Read A Book” to “Knowledge Gained?”

4. Create a Flow Relation from your Condition to a task.

Set the Flow Relation from “Knowledge Gained?” Condition to “Feel Smarter” Atomic Task.

5. Create another Flow Relation from your Condition to another task to signify the two possible flows from the Condition.

Before we create our second flow relation from our Condition, create another Atomic Task and call it “Look for Another Book”.

Change the Join Decoration for Atomic Task “Read a Book” to an OR Join, with the orientation being West, by right clicking on Read a Book and choosing **Decorate**. Click **Done**.

Finally create the Flow Relation from the “Knowledge Gained?” Condition back to the OR Join of the “Read A Book” Atomic Task, as per Fig. 13.

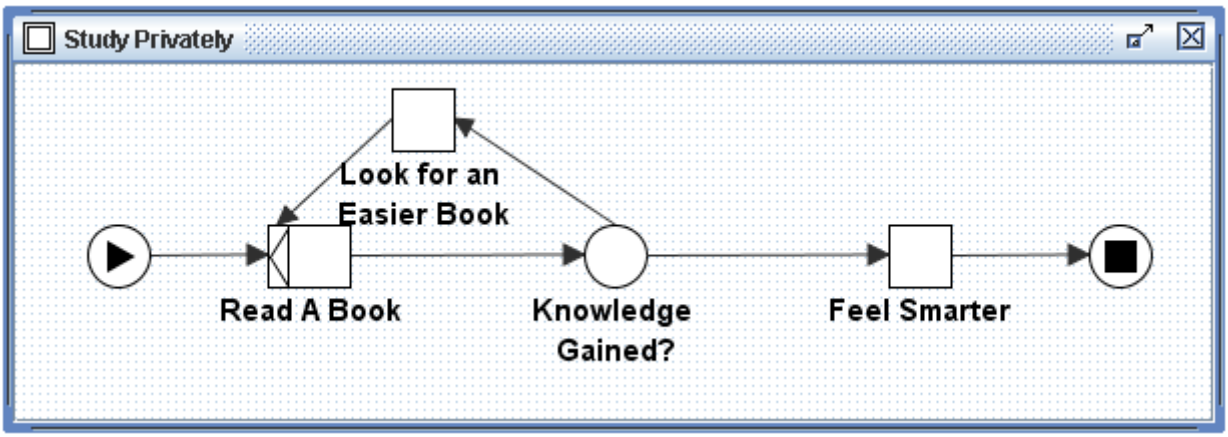


Fig. 13 Condition Loop for “Study Privately” Net

6. Validate your specification. Validation should fail and report errors as per Fig. 14. The problem here is that the “Study Privately” multiple instance composite task needs to have more information specified for it to be valid.

For setting data detail of multiple-instance tasks, please see the “Setting Multiple Instance Queries” section of this manual.

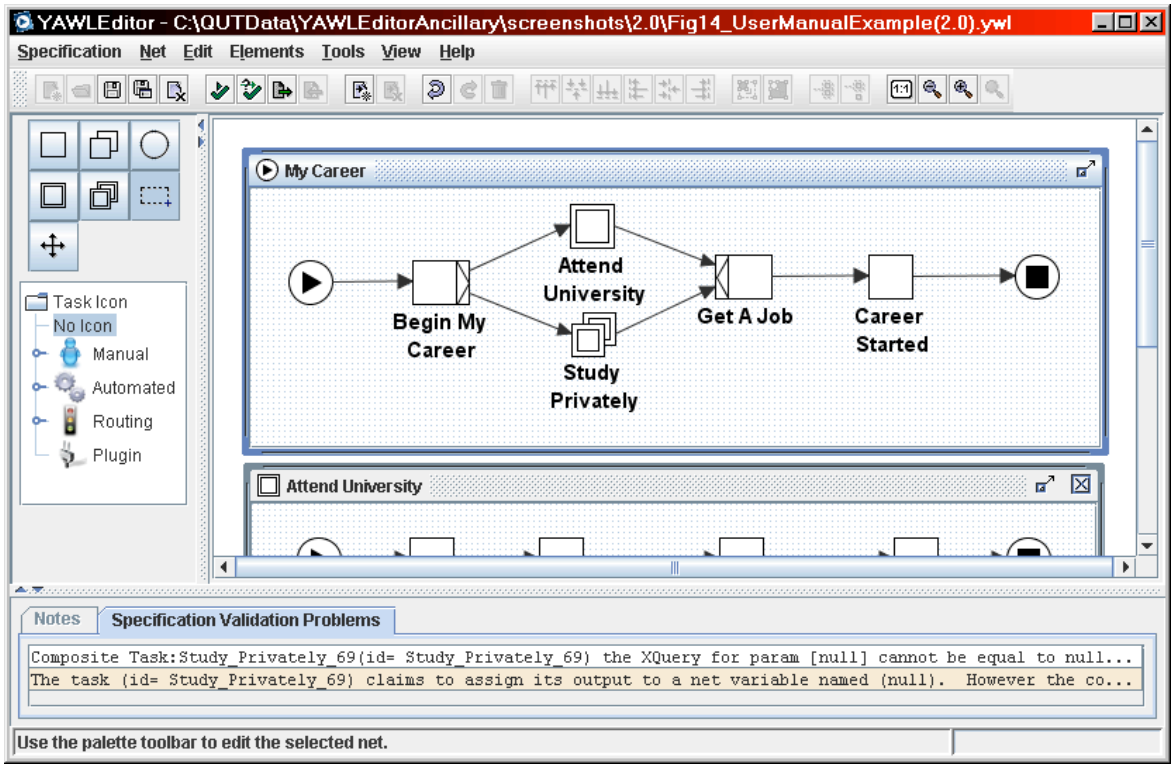


Fig. 14 Validation with unfinished Multiple-Instance Tasks

7. Remove the ''Study Privately'' multiple-instance composite task and replace it with an atomic composite task using the same decomposition, and re-drawing flows from “Begin My Career” and to “Get A Job” tasks. Your updated net should look like Fig. 15.

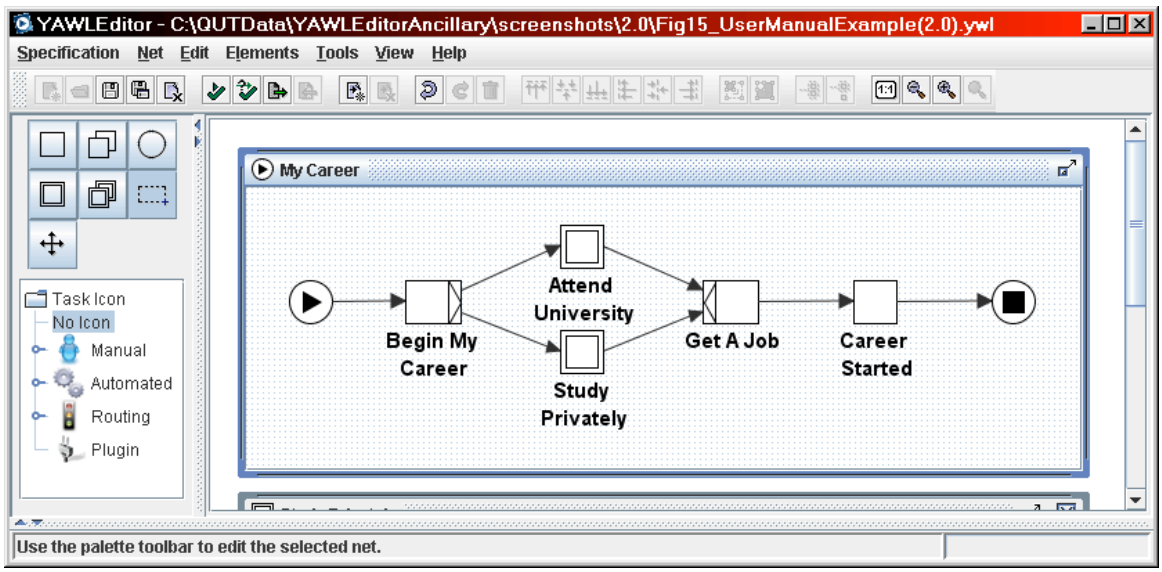


Fig. 15 Making the "Study Privately" task an atomic composite

Changing the Starting Net

At any stage you can change the starting Net of the specification. To change the starting Net:

1. Select **Net** from the Menu.
2. Choose **Set Starting Net** (Fig. 16).
3. From the Choose Starting Net window, click on the drop-down list and select a new starting Net.
4. Click **Done**.

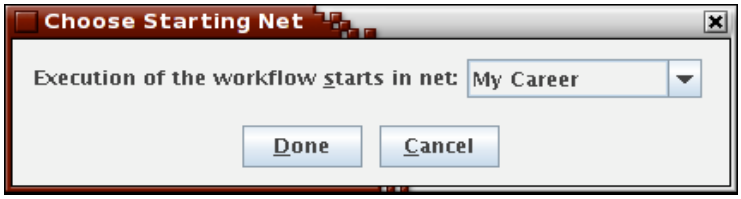


Fig. 16 Changing the Starting Net

Note that the starting net has an input condition symbol, ▶, in its title frame, and its minimised icon. All sub-nets have a composite task symbol, ▣, in their title frames and minimised icons.

Changing the Appearance of Your Specification

Improving the Look of Flow Relations

You can control and improve the look of the Flow Relations by adding in “knees”.

1. Select the Enter Marquee Selection Mode button, , from the Palette Bar.

Go to the “Study Privately” Net.

2. Right click on the position in the Flow Relation where you want to add a “knee”, signified by a small square, , in the Flow Relation. A popup menu will appear, allowing you to add and remove knees, as well as change the line style of the flow.

Create a knee somewhere on the Flow Relation going from “Knowledge Gained?” Condition to “Look for an Easier Book” Atomic Task. Then left click on the knee created and drag it out to a more desirable location. You can add as many knees to a Flow Relation as you like.

3. Repeat the process for the Flow Relation between “Look For an Easier Book” and “Read A Book” tasks (see Fig. 17).

   You can now reconnect flow relations to other elements of a net, or different points on the same element by selecting the flow, and dragging one of its connecting ends from one net element to another. If a connection is possible at some other element, connection points will become visible as described earlier. Release the mouse button to attach the flow to its new home.

4. Take the current flow relation, and move it from the top of the task to its side, as depicted in Fig. 17.

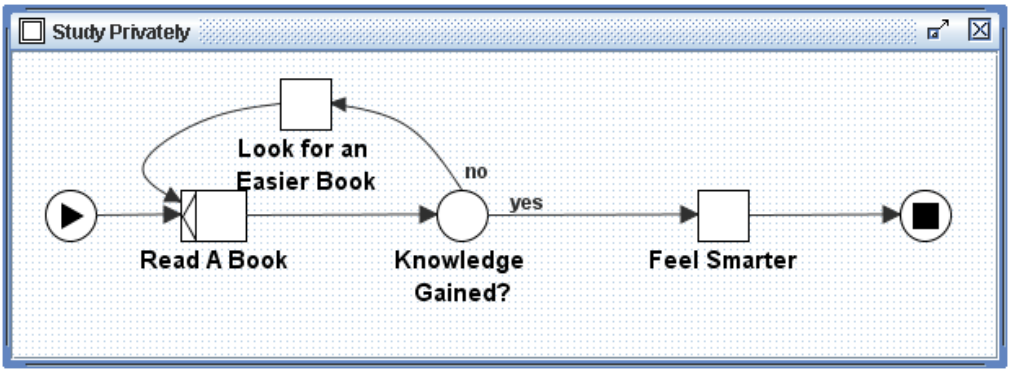


Fig. 17 Adding "knees" to a Flow Relation

It is also possible to add labels to flows. To do so, double click on a flow when the pallet button “Net Element Selection Mode”, , is selected. A small editor will appear over the flow. Type your desired text, and commit the flow label by pressing the ENTER key. You may then drag that flow label around as desired.

5. Take the two flow relations that have recently had knees added to them. Attach the label “yes” to the flow relation going from the “Knowledge Gained?” condition to the “Feel Smarter” atomic task. Attach the label “no” to the flow relation going from the “Knowledge Gained?” condition to the “Look for an Easier Book” atomic task. Drag the labels about to a desired position, much like what has been done in Fig. 17.

   Note that Fig. 17 shows flows using two different line styles. The flow running from “Look for an Easier Book” has been given the “spline” line style in this figure, while the remaining flows are all “orthogonal”, resulting in sharp edged knees on flows, such as the one running from the “Knowledge Gained?” condition to the “Look for an Easier Book” task.

Editing Objects

You can edit more than one object at a time by using the Enter Marquee Selection Mode tool. See Fig. 18.

1. Select the Enter Marquee Selection Mode button, , from the Palette Bar.

2. Click on the first object that you want to edit.

3. Hold down the shift key and then click on the other objects that you want to edit.

4. Now choose the **Edit** option from the Menu or continue holding down the shift key and right click on the mouse button. Below are the edit options:

   - Cut, Copy, Delete
   - Align
   - Size Increase / Decrease to change the appearance of the objects. This can also be done using the CTRL key plus Up or Down arrow on your keyboard.

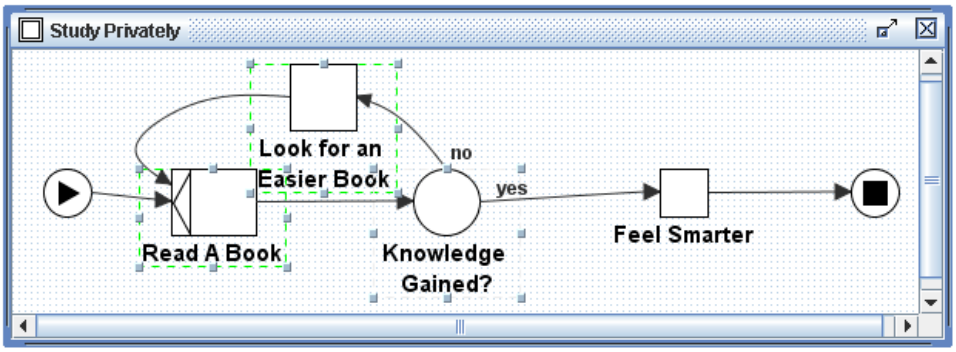


Fig. 18 Changing the Size of Multiple Objects

Note also, that whenever you have selected a number of net elements, pressing one of the arrow keys will move the selected elements in the direction of the arrow key, and pressing the CTRL key plus the A key will select all elements in the net with input focus.

Changing Font Size

You can change the size of the font used to label tasks and conditions.

1. Change the font size by clicking on the **View** Menu and choosing the **Label Font Size…** option.

   Change the font size to 18.

Changing Task Icons

You can change the icon that renders for atomic tasks.

1. Select any single atomic task in your workflow. The palette will expand to include a task icon tree, depicted in Fig. 15, where you can an icon from the tree to the task. You are free to assign any icon. Icons have no runtime effect on the engine, and are provided simply to make specifications more easily understood by people observing the specification in the editor.

Using Custom Icons

Workflow designers can plug in and use their own icons for specification design. Icons must be of the PNG file format, and be a maximum of 24x24 pixels to render properly within editor task boundaries.

When the editor starts, it checks the installation folder for a plugin directory. Currently it looks in the directory

```
<editor_installation_path>/YAWLEditorPlugins/TaskIcons
```

for user-defined icons, and adds them into the plugin branch of the task icon tree widget of the editor's palette. Sub-directories are supported, and will form new sub-trees of the same name when the plugin sub-tree is being created. If an icon cannot be found that was previously used for a specification, a special “broken” icon will render in its place, as depicted in Fig. 19.

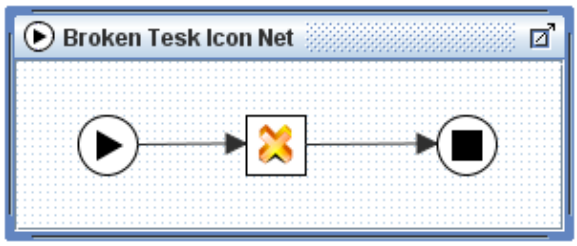


Fig. 19: A task specifying an icon that the editor cannot locate

Advanced Specification Features

Cancellation Sets

Cancellation Sets allow you to nominate any number of tasks, conditions or implicit conditions (which exist on flow relations between tasks but are not drawn) for cancellation, upon the execution of a specified task. That is, once a specified task is executed in the workflow, all state within that task's cancellation set is removed.

To create a Cancellation Set:

1. First select the task that will initiate the Cancellation Set, using the Enter Marquee Selection Mode button, , from the Palette Bar.

2. Right-click on the task, then choose **View Cancellation Set**.

   The task will be coloured grey to indicate that this task is the cancellable task.

Go to the "My Career" Net. We are going to make the "Attend University" task a cancellable task to demonstrate in the workflow, that if a person executes this task, then they don't have time to perform the "Study Privately" task, therefore it should be removed as an option.

Right-click on "Attend University" and choose **View Cancellation Set** (as per Fig. 20).

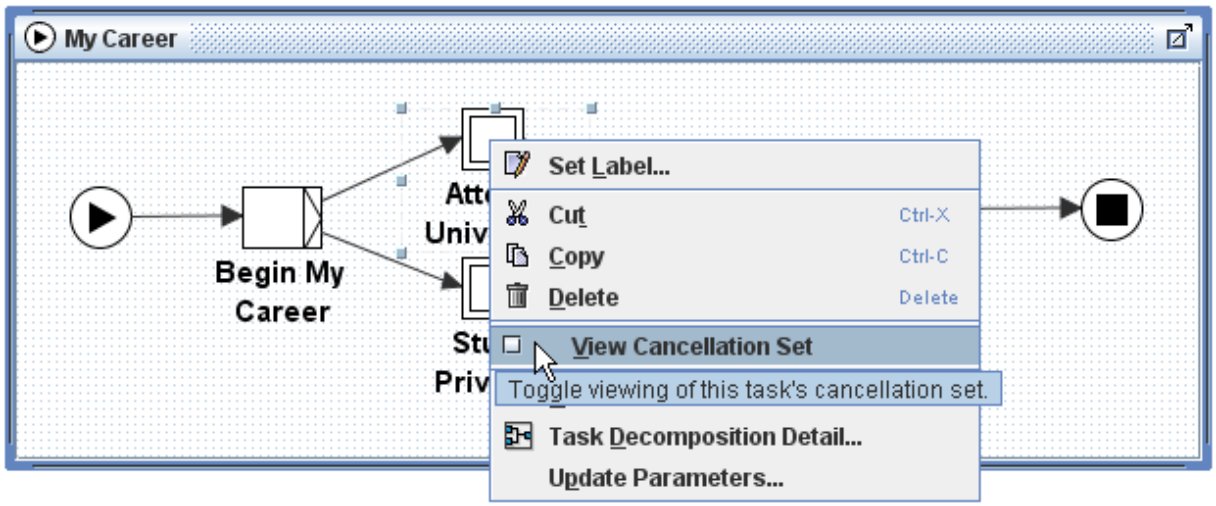


Fig. 20 Viewing a task's cancellation set

3. Next, using the Enter Marquee Selection Mode button, , from the Palette Bar, choose a task, condition or implicit condition to add to the Cancellation set. Hold down the shift key to select more than one object for cancellation.

Select the “Study Privately” task and the Flow Relation preceding it.

4. Click on the Add Selected Items to Visible Cancellation Set button, , on the Menu Toolbar.

   Items will be coloured red to indicate belonging to the cancellation set (see Fig. 21).

Add the “Study Privately” task and the preceding Flow Relation to the cancellation set.

5. Once you have established the cancellation set, you can right-click on the cancellable task and tick-off **View Cancellation Set** option.

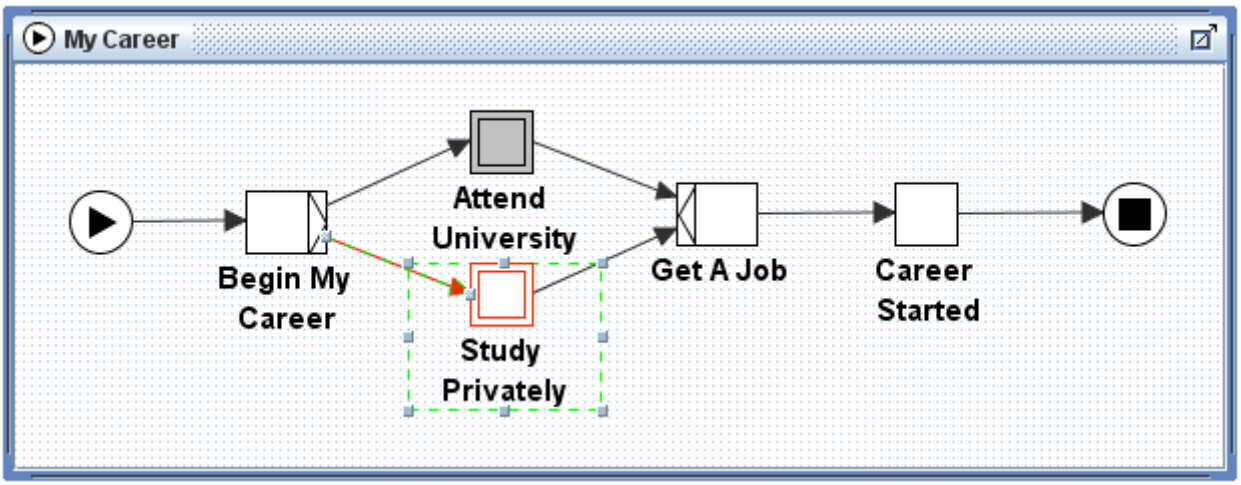


Fig. 21 Adding elements to a Cancellation Set

Notes about Cancellation Sets:

- A Cancellation Set that has been created will remain in the workflow, regardless of whether you have the **View Cancellation Set** option ticked.
- You can create multiple Cancellation Sets in your workflow, by selecting another task and choosing the **View Cancellation Set** option.
- All flows leading to or from conditions are not valid cancellation set members. Neither are the Input and Output conditions. The editor will ignore them if you select them for inclusion in a task’s cancellation set.

To remove an element from a Cancellation Set:

1. First, make sure you have the **View Cancellation Set** option ticked.

   If it isn’t ticked, select the task that is initiating the Cancellation Set, using the Enter Marquee Selection Mode tool from the Palette Bar and right-click, then choose **View Cancellation Set**.

2. Select the element for removal, using the Enter Marquee Selection Mode button, .

3. Click on the Remove Selected Items from Visible Cancellation Set button, , on the Menu Toolbar.

Data Type Definitions

Version 1.1 of the YAWL Editor introduced the ability for workflow designers to specify data that is processed by a workflow. XML Schema is used to describe the valid data types that are available.

By default, a number of simple XML Schema data types are supplied for variable definition, but if you need more complex data types, you can supply your own XML Schema definition to describe them.

1. Select **Update Data Type Definitions** from the **Specification** Menu.

2. Enter your XML Schema Data Type Definition into the dialog box. (See Fig. 22).

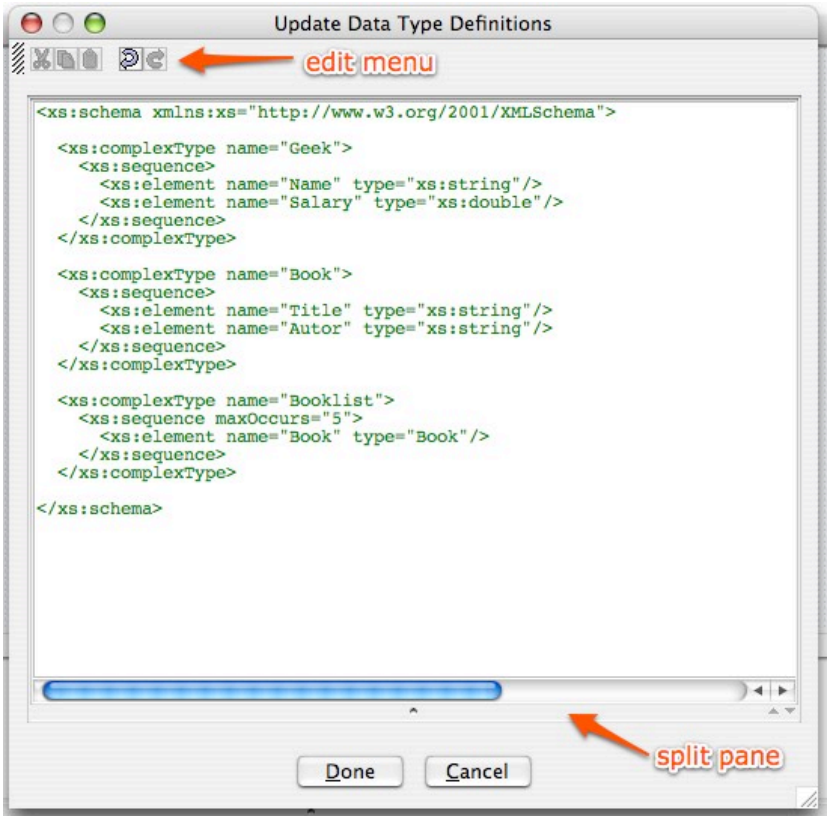


Fig. 22 Adding the "Geek" complex data type

3. If the definition text is green, your new data type is now available for defining Net or Task variables in your specification. If the text is red, there is something wrong with your data type definition, and the data type will not be available.

When the text is red, the split-pane will reveal a table listing parse errors that were collected when determining the validity of the text supplied. An example of this is shown in Fig. 23.

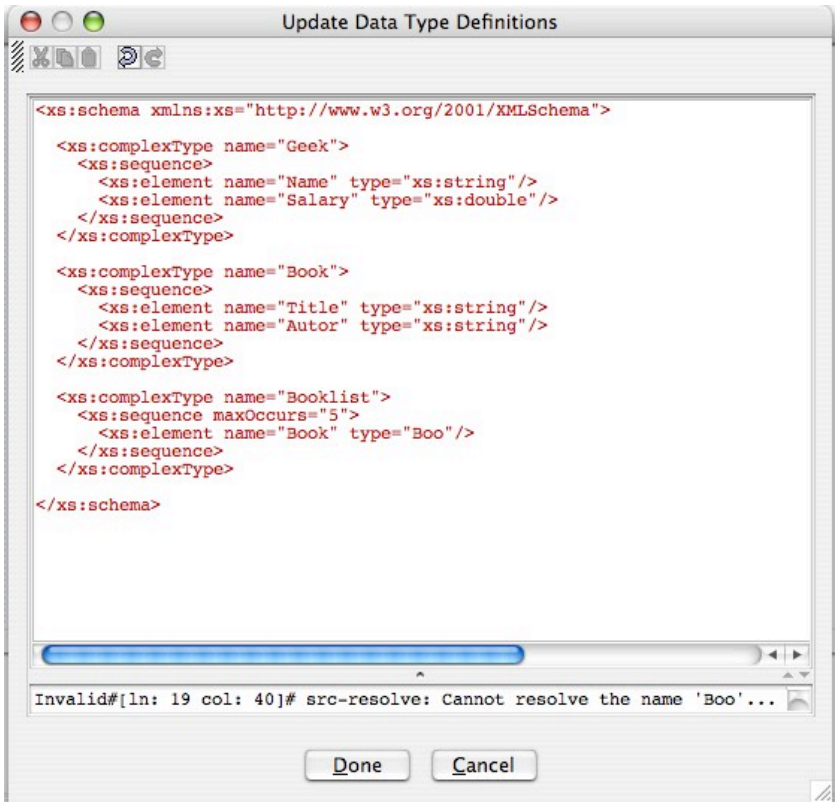


Fig. 23 When the data type definition is invalid

4. Open the Data Type Definitions dialog and type in the XML text that appears in Fig. 22.

The above example creates a complex data type called "Geek" that has two separate sub-components, "Name" and "Salary" of type "string" and "double" respectively. As depicted in Fig. 24, the new data type "Geek" is available to choose when creating a task or net variable. Variables with a usage of "Local" can have initial values specified for them, as depicted in the same figure. As with the data type definition dialog, parse errors will be listed when the initial value text is red.

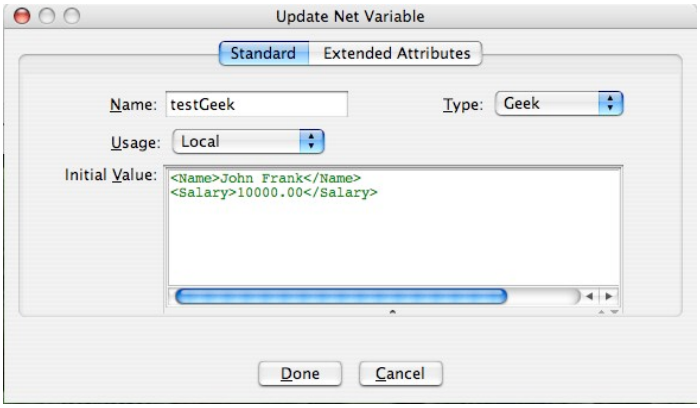


Fig. 24 A "Geek" net variable with initial value

Net Decomposition Detail / Net Variables

You can add variables to a net to store information relating to that net that tasks within the net may need to read or update.

To add a variable to a Net:

1. Choose **Update Net Detail** from the **Net** Menu.

We will be setting up Net variables in the “Attend University” net.

Go to the “Attend University Net” and choose **Update Net Detail** from the **Net** Menu.

2. An Update Variables dialog box will appear (see Fig. 25). Click on **Create**. An Update Net Variable dialog box will appear. Click **Done**. Create another Net Variable with the name “SubjectCode” and Type of **string**. Leave the Initial Value blank and set the Usage to “Local”. Click **Done**.

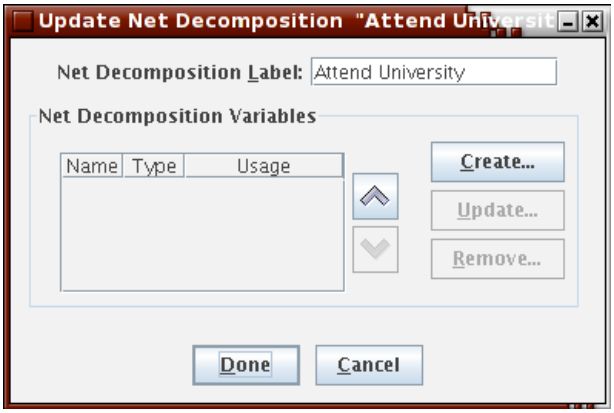


Fig. 25 Updating "Attend University" Net Variables

3. Enter the Name of your variable, choose the Type of the variable from the list, then click **Done**, then **Done** again to close the Net Decomposition dialog.

Type in “StudentNumber” for the name of the variable and choose its type as **string**. Set the Usage to “Local” (Fig. 26).

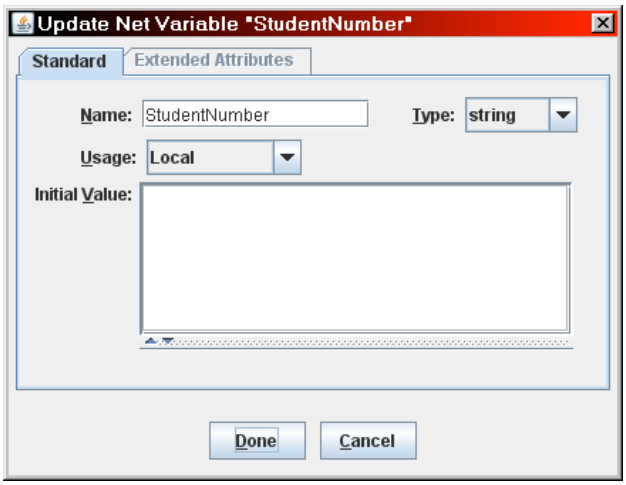


Fig. 26 The Net Variable "Student Number"

4. The Net Variables should now appear in the Update variables of Net “Attend University” dialog box (Fig. 27).

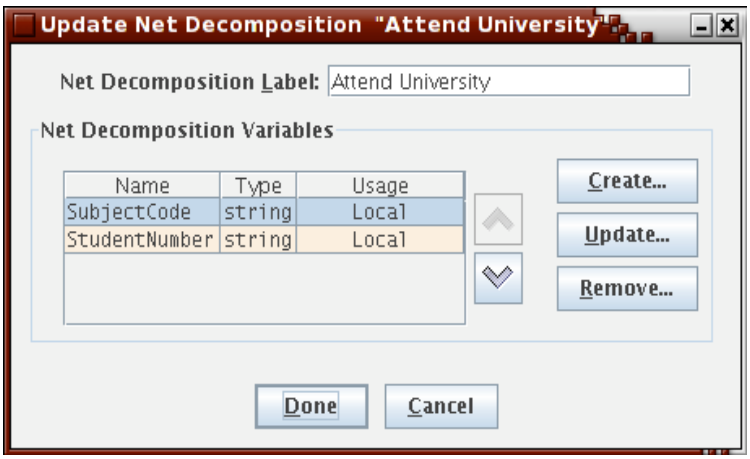


Fig. 27 Updated "Attend University" Net Variables

Task Decomposition

By choosing the **Select Task Decomposition…** option when you right click on a task, you have the ability to identify which decomposition this task represents.

Like nets, tasks have decompositions where you can specify variables and a label to associate with the task or net. Unlike nets, which cannot share net decompositions,

there is a 1:N relationship between task decompositions (scoped to the entire specification) and their tasks (scoped to nets).

Besides variables and a label, task decompositions also allow the workflow designer to identify which web service the decomposition should invoke in a running workflow engine, and whether the decomposition will create manual (i.e. human-actioned) or automated (non-human-actioned) tasks. When two tasks share the same decomposition, we are saying that the same activity is required in two different places in the workflow.

You can use the drop-down list to select the task decomposition of a task, or alternately you can press the **Create…** button and generate a new one that will automatically become the task's decomposition (Fig. 28).

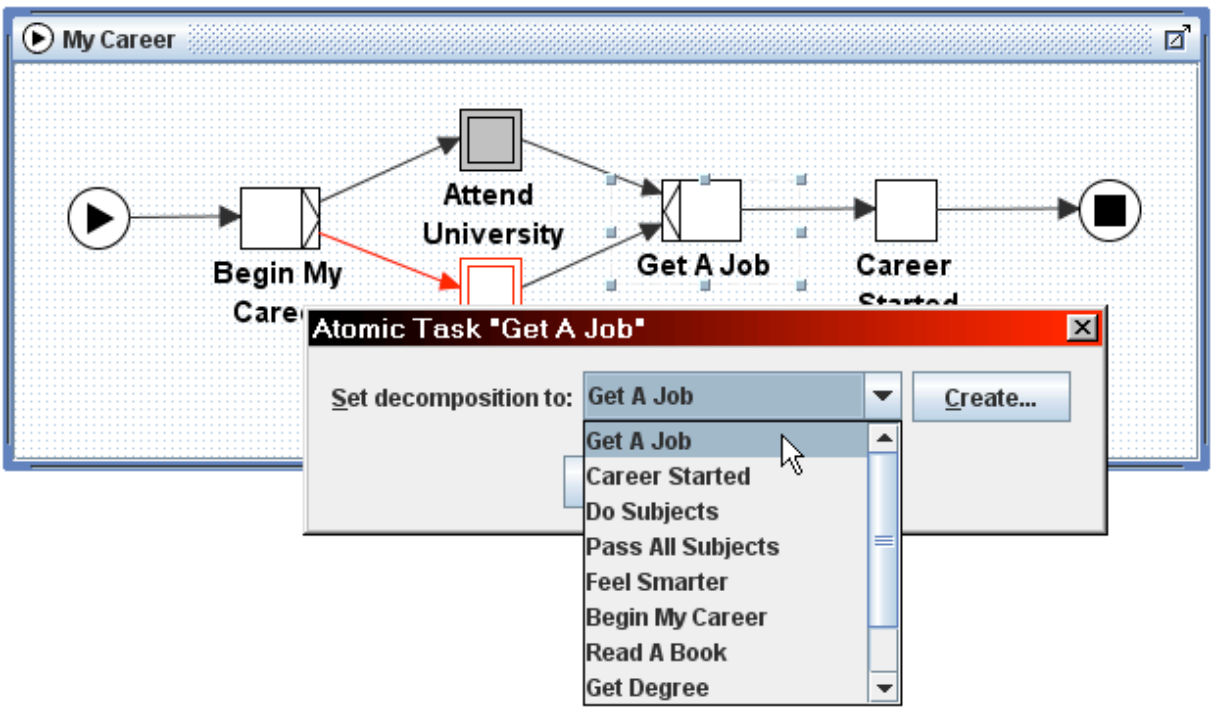


Fig. 28 Select Task Decomposition

Task Decomposition Detail / Task Variables

You can add variables to a task to store specific information relating to that task. Task variables have several uses. One use is to transferring information between workflow users and the workflow engine. A second use is for passing data between web services and/or external code and/or applications that the running workflow engine invokes and the Net the task resides in.

For example, if your task is called 'Place an Order', you may want to store the name or identification number of the person placing the order or maybe store the items being ordered.

Adding a Variable to a Task

1. First select the task that will require the variable, using the Enter Marquee Selection Mode button, , from the Palette Bar.

We will be setting up variables for the “Enrol” task.

Go to the “Attend University” Net and select the “Enrol” task.

2. Right-click on the task and choose **Task Decomposition Detail…**. An Update Task Decomposition dialog box will appear (Fig. 29).

Retrieve the decomposition detail for the “Enrol” task.

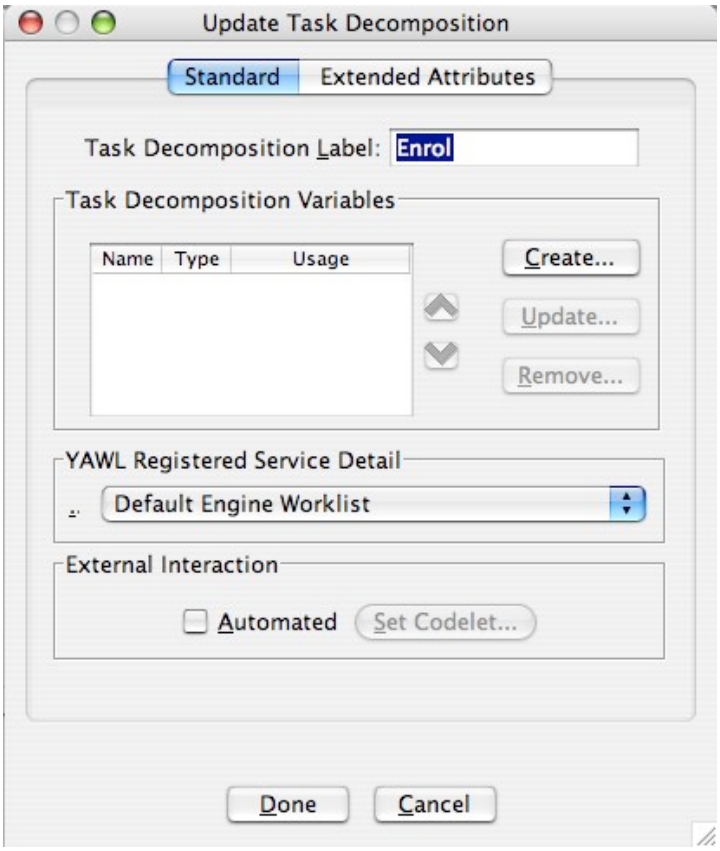


Fig. 29 Updating variables for a task

3. Click on the **Create…** button. An Update Task Variable dialog box will appear.

4. Enter the Name of your variable, choose the Type of the variable from the list, click **Done**, then **Done** again to exit the task decomposition detail dialog.

Type in “StudentNumber” for the name of the variable and choose the type **string** (Fig. 30).

5. Create another variable for the same task, called “SubjectCode” with the type being **string** and usage being Input & Output**.** Click **Done** (Fig. 31).

6. The “Enroll” task now has two variables, “StudentNumber” and “SubjectCode”.

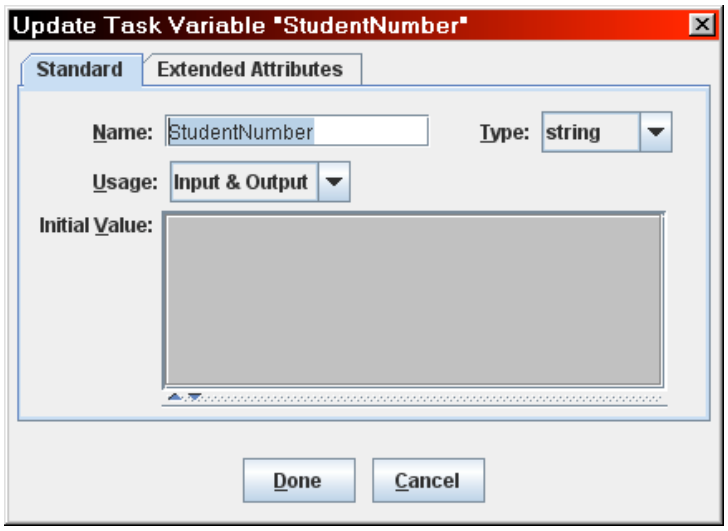


Fig. 30 Updating the Task Variable

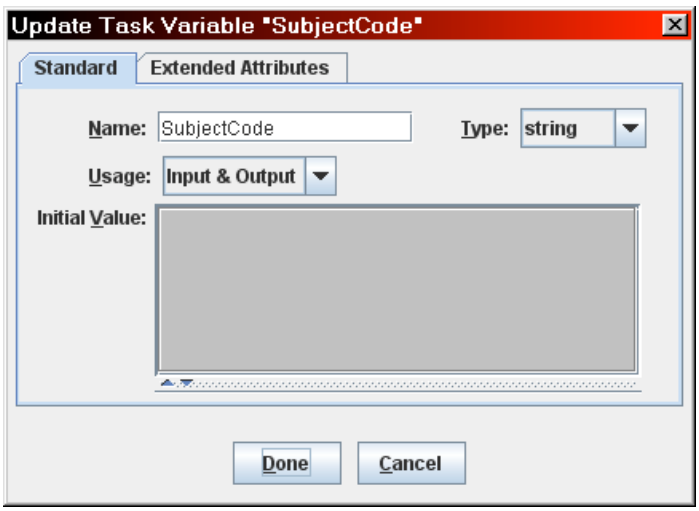


Fig. 31 Updating another Task Variable

Task Parameters

Both Input and Output Parameters can be assigned to any tasks to allow the passing of state between nets and their tasks, and between tasks and workflow engine users and web services.

Input Parameters use an XQuery to massage net variable state (across possibly several net variables) into a value that can be passed to a single selected task variable.

Output parameters use an XQuery to massage task variable state (across possibly several task variables) into a value that can be passed to a single selected net variable.

For example, if a task is called 'Place an Order', then an Input Parameter could be the name of the person placing the order, whereas the Output Parameter could be the corresponding identification number of that person.

To add an Input Parameter:

1. Select the task that will require the parameter, using the Enter Marquee Selection Mode button, , from the Palette Bar.

   We will be setting up Input Parameters for the variables that we created in the Adding / Updating Task Variables section previously.

   Go to the "Attend University" Net and select the "Enrol" task.

2. Right-click on the task and choose **Update Parameters…**. An Update Parameters dialog box will appear (**Error! Reference source not found.**).

   Update the Parameters for the "Enroll" task.

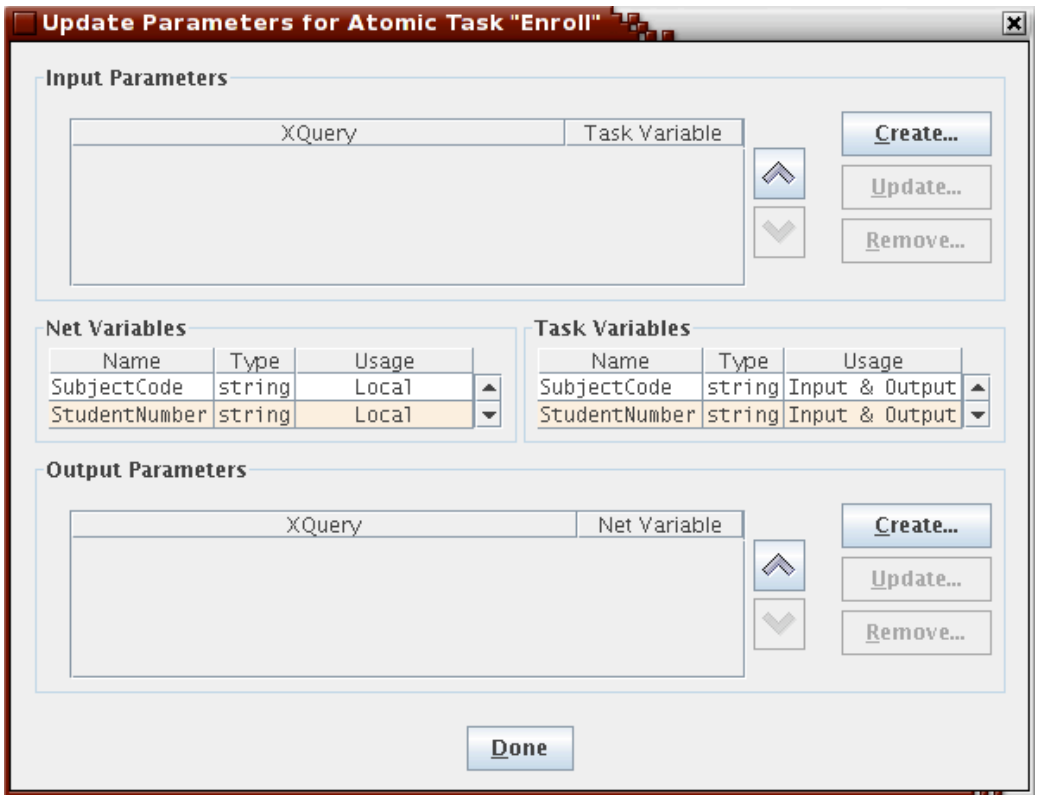


Fig. 32 Updating Parameters for a Task

3. In the Input Parameters section, click on **Create…**. An Update Task Parameter dialog box will appear.

   If you have already set up a Task Variable for this task, then the Existing Task Variable option will be activated and there will be a list of task variables to choose from. Choose a variable from the list and click on **Done,** then **Done** again to close the task parameters dialog.

   If you haven't set up Task Variables, then click on **Create…** and return to the previous section in the manual for Task Variable.

   If you are familiar with XQuery syntax, then you can paste in an XQuery to allow manipulation of the Input Parameters. "Syntactically well formed" XQueries will be green, and badly formed ones will be red. Again, red text will be accompanied by a split-pane table, returning the parse errors that cause the text to be badly formed.

   For workflow designer convenience, two XQuery buttons are supplied to generate XQuery expressions from available net variables.

   The **add XQuery of element's content only** button will return just the content of the XML element for this variable, which is useful for simple state transfer between two variables of the same XML Schema type, and expected to be the typical button that users will start out with.

   The other button, **add XQuery of entire element**, will return the entire XML element of the selected variable, which is useful for times when you want to create a complex type expression from individual variable elements. Experience with XMSchema and XQuery are necessary to understand the effects this button will have on runtime YAWL engine state.

   Select the "SubjectCode" variable from the list of Existing Task Variables. Select the "SubjectCode" net variable and click **add XQuery of element's content only** (Fig. 33) Click **Done.**

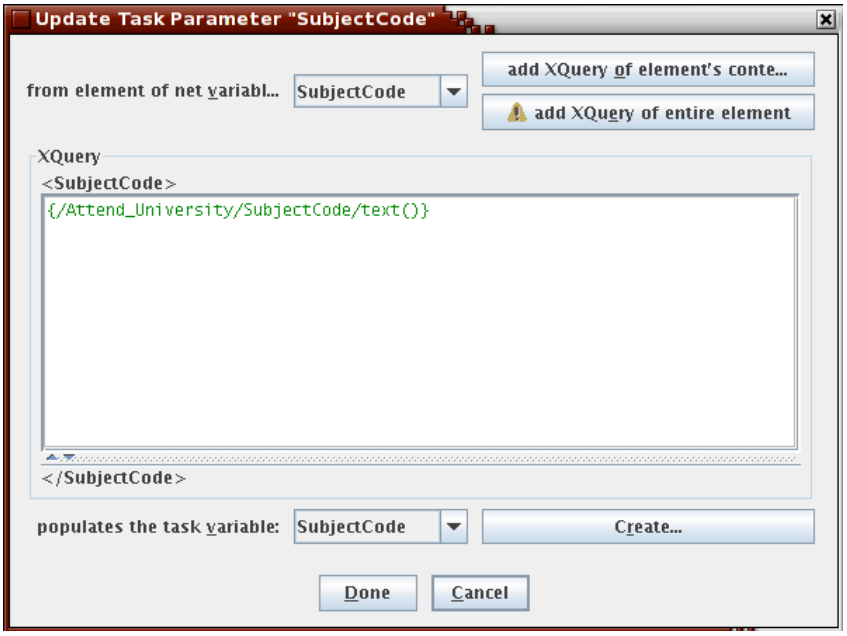


Fig. 33 Passing a net's SubjectCode value to a Task

Create another Task Parameter and map the net "StudentNumber" to the task variable of the same name using this technique. Click **Done** and **Done** again.

To add an Output Parameter:

1. First select the task that will require the parameter, using Enter Marquee Selection Mode button, , from the Palette Bar.

   We will be setting up Output Parameters for the variables that we created in the Adding / Updating Task Variables section.

   Go to the "Attend University" Net and select the "Enrol" task.

2. Right-click on the task and choose **Update Parameters…**. An Update Parameters dialog box will appear (Fig. 33).

   Update the Parameters for the "Enrol" task.

3. In the Output Parameters section, click on **Create**. An Update Net Parameter dialog box will appear (similar to Fig. 33).

4. If you have already set up a Task Variable for this task, then the Existing Task Variable option will be activated and there will be a list of task variables to choose from. Choose a variable from the list and click on **Done,** then **Done** again to close the task parameters dialog.

   If you haven't set up Net Variables, then click on **Create…** and return to the previous section in the manual for Adding a Net Variable.

If you are familiar with XQuery syntax, then you can paste in an XQuery to allow manipulation of the Output Parameters. "Syntactically well formed" XQueries will be green, and badly formed ones will be red.

For workflow designer convenience, two XQuery buttons are supplied to generate XQuery expressions from available task variables.

The **add XQuery of element's content only** button will return just the content of the XML element for this variable, which is useful for simple state transfer between two variables of the same XML Schema type, and expected to be the typical button that users will start out with.

The other button, **add XQuery of entire element**, will return the entire XML element of the selected variable, which is useful for times when you want to create a complex type expression from individual variable elements. Experience with XMSchema and XQuery are necessary to understand the effects this button will have on runtime YAWL engine state.

5. From the list of task variables, select the "SubjectCode" task variable and click **add XQuery of element's content only**. From the list of net variables, select the "SubjectCode" variable. Click **Done**.

   Create another Task Parameter and map the task "StudentNumber" to the net variable of the same name using this technique. Click **Done** and **Done** again.

   These Output Parameters were set up to demonstrate a simple transfer of state from a net to a task and back to the net. Perhaps the task would allow a user to change the values of one of the variables which would eventuate in the net's values changing.

   The Update Parameters dialog box should appear as in Fig. 34.

A brief introduction to XQuery

XQuery is a separate language that describes how to manipulate a given XML document to extract data from it. For example, the XQuery:

```
/data/myTaskVariable/number() + 5
```

is saying that there is some XML Schema element in an engine XML document with an element called `myTaskVariable` whose content is a number, and that the resultant XQuery expression is that element's value with 5 added to it.

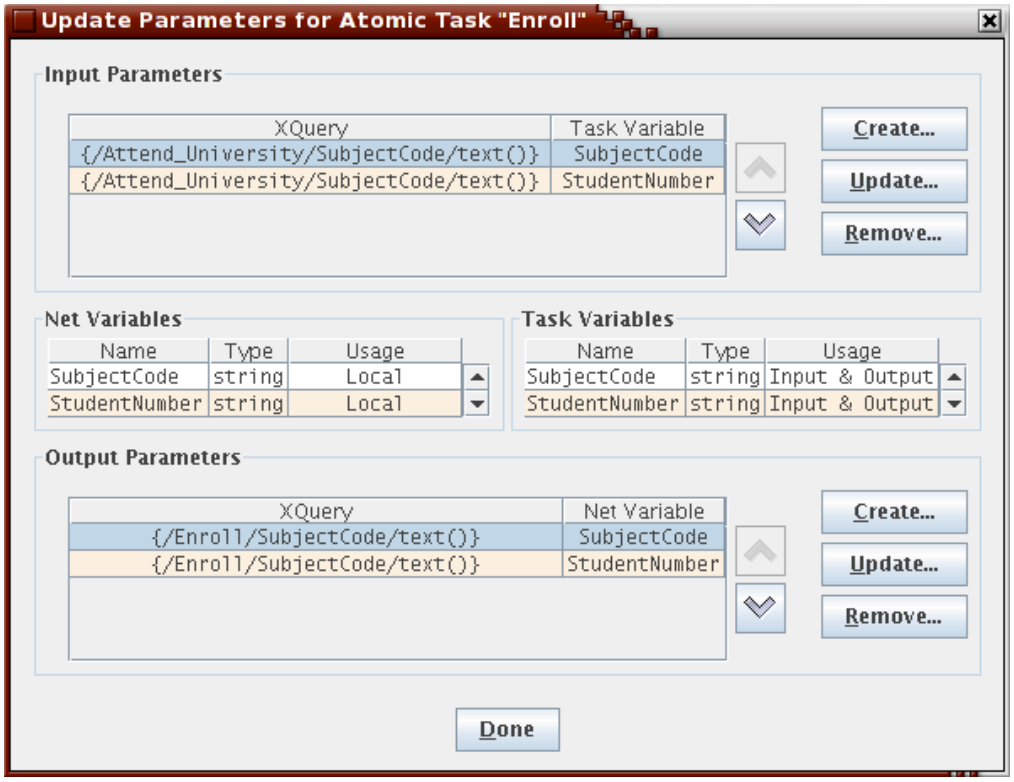


Fig. 34: Established task parameters

Flow Detail

When dealing with tasks that have XOR and OR splits, we need some way of telling a running engine which flows should be activated. By updating the flow detail, you can specify the default flow path and also establish predicates for each flow direction. Flows whose predicates evaluate to true in a running engine will execute those flows.

To update the flow detail of a task that has a split:

Right click on the task and choose **Update Flow Detail…**.

The number of splits will be the number of Target Tasks. The currently selected flow in the dialog will also be identified by being drawn green in the Net (Fig. 35).

To choose the flow ordering, select the Target Task that you want as the change the order of, use the Up and Down arrows on the right to position the flow. The bottom-most flow will be used as the default if some flow needs to be followed, but no predicates evaluate to true.

To specify a predicate for a particular flow, select the flow from the list and click on **Predicate…**

Enter a predicate as a boolean XQuery expression and choose **Done**.

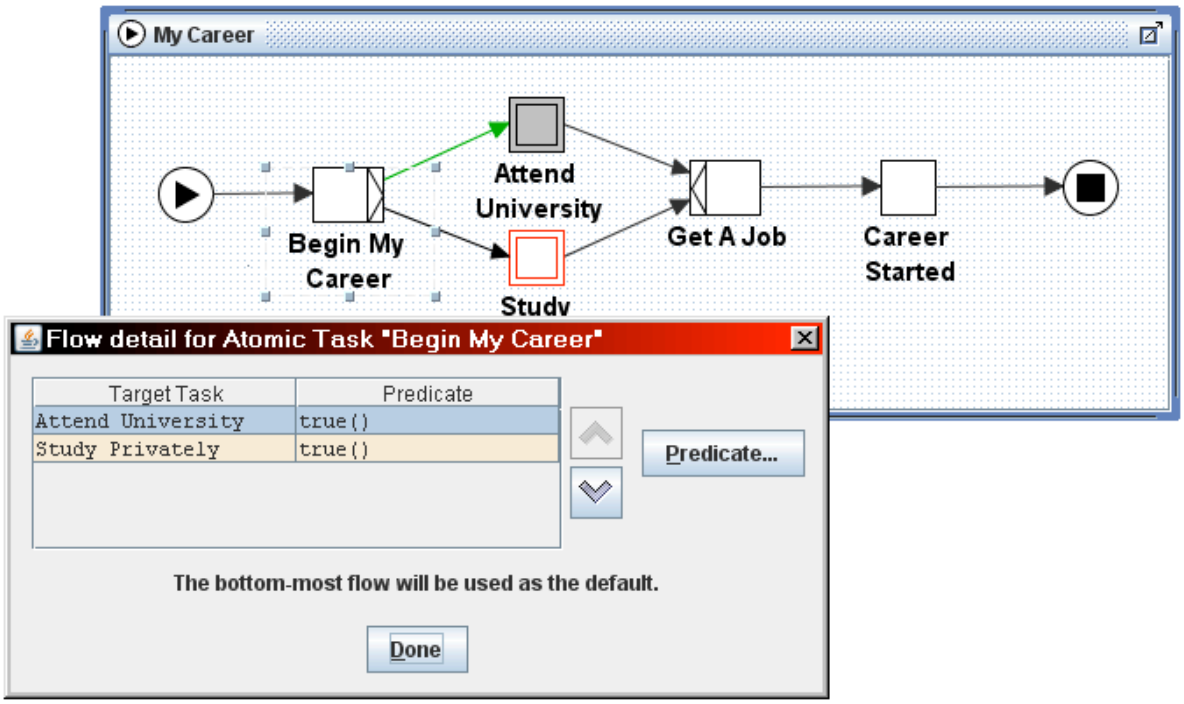


Fig. 35 Updating the Flow Detail

Multiple Instance Queries

Multiple Instance Tasks need extra detail for data manipulation. The “Queries” tab, available by right-clicking on a multiple-instance task and selecting **Set Instance Detail…**. The “Queries” tab allows you to manipulate multiple instance data for the task.

Right click on the task and choose. Ensure that you are viewing the “Queries” tab of the dialog, as per Fig. 36.

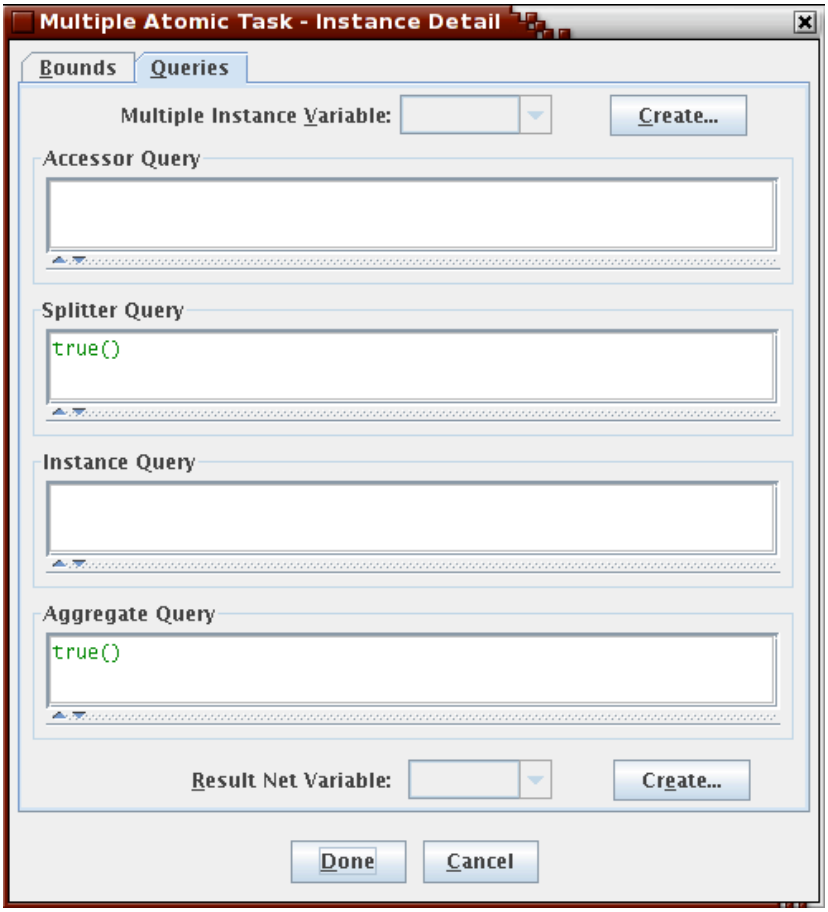


Fig. 36: Data manipulation for Multiple Instance Tasks

To successfully specify multiple instance data manipulation, you first need to select which of the task's decomposition variables will be used as the "Multiple Instance Variable". This variable needs to contain (or can derive) a number of unique values that will be separated with the Splitter query to pass a unique value per each instance task. If the "Multiple Instance Variable" needs manipulation overall before the unique values are split out of it, use the Accessor Query to do so.

On the completion of an instance, the "Instance Query" can be used to transform the XML document returned to a form suitable for the Aggregate query to finally generate an overall result. The overall result will be assigned to the "Result Net Variable" on completion of the multiple instance task.

See Fig. 53 for an example of this screen being filled out with valid XQuery expressions.

Fast-Tracking Data Definition

Workflow designers have an option to directly transfer data from the net to a task's variable and then back again. This can be done by right-clicking on an atomic task that currently does not have a decomposition, and selecting **Decompose to Direct Data Transfer**. The dialog in Fig. 37 will pop up.

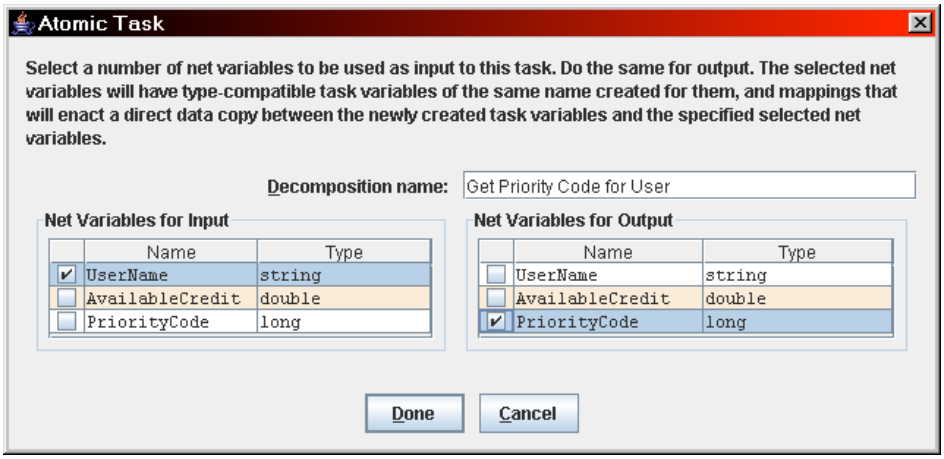


Fig. 37 An example of the direct data transfer dialog

This dialog will automatically create a task decomposition and matching XQueries to directly transfer data from a selected net variable to a task and back again.

End of Scenario

This is where our scenario ends. The My Career Scenario was designed to explain all the functions of the YAWL Editor and to provide you with a rudimentary understanding of designing a YAWL workflow specification.

But the complexity of the YAWL Editor does not stop with the current scenario. If you are after something more challenging, try adjusting your version of the scenario to expand into more sub-Nets and more complex situations.

The completed example described in this scenario is supplied with the editor source distribution (Fig. 38).

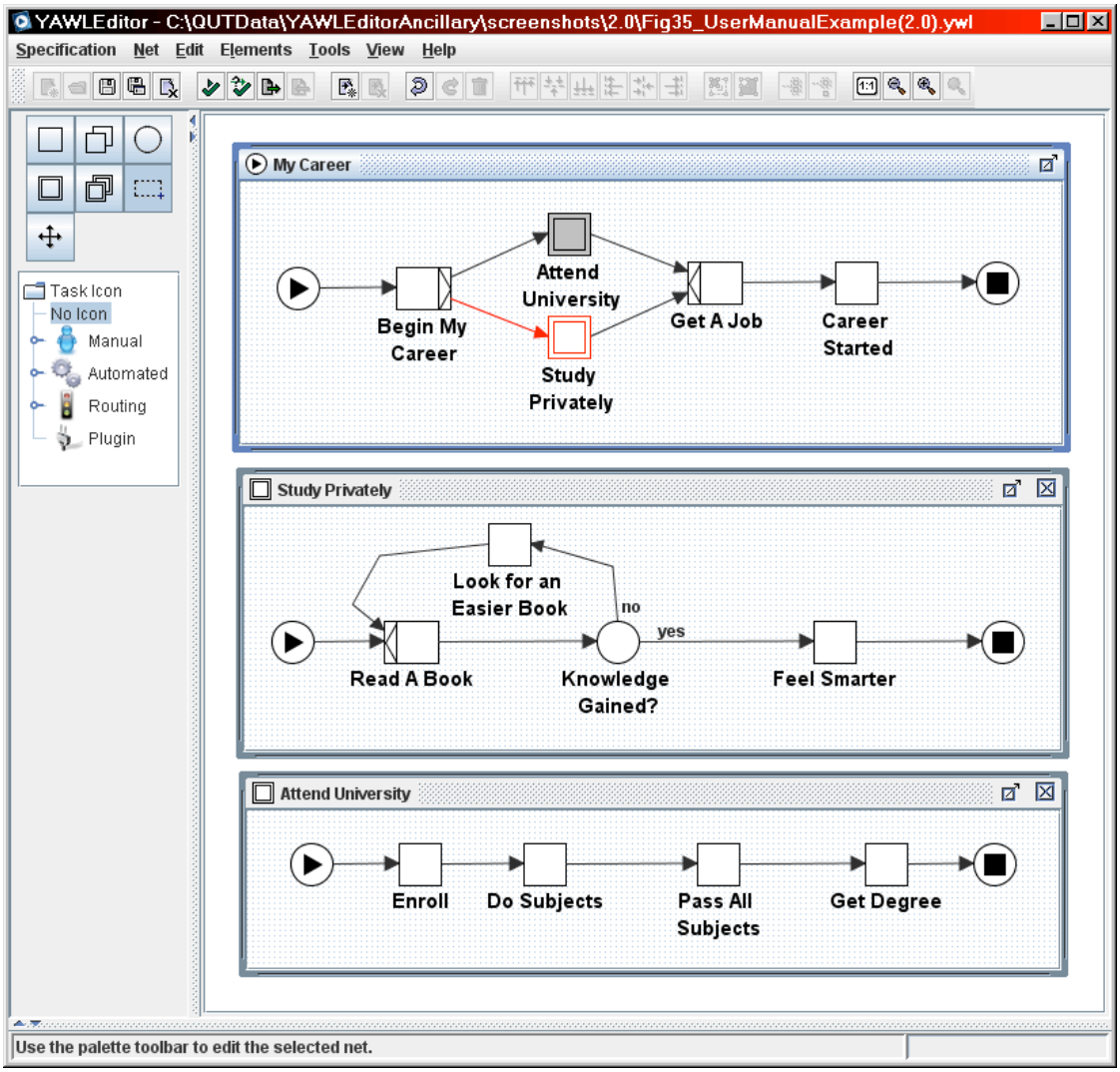


Fig. 38: The complete scenario specification

Connections

Connecting to the YAWL Engine

In order to generate specifications that can be executed by the YAWL Engine, it is required to establish a connection between the Editor and the Engine.

The Editor receives from the Engine a list of the current web services that the Engine has been configured to interact with. Further detail on specifying web services to have the workflow interact with can be found in the next section.

To set establish connection with the YAWL Engine:

1. Click the **Tools** Menu and choose the **Engine Connection Settings…** item.
2. From the resulting dialog (Fig. 39), accept the default values or enter the following engine details:
   - YAWL Engine URI
   - Administrator's User Name
   - Administrator's Password.

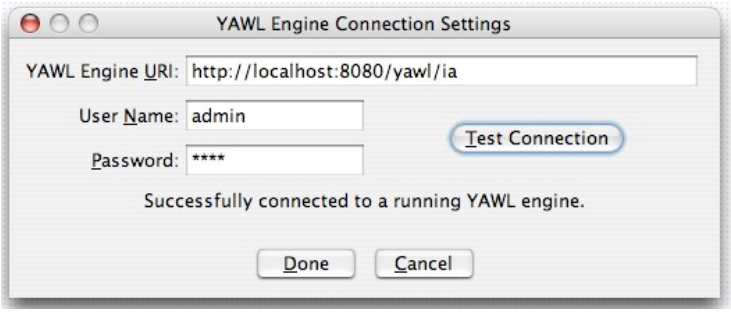


Fig. 39 Specifying the YAWL Engine connection detail

The YAWL Engine URI value is set by default to a locally installed Engine (`http://localhost:8080/yawl/ia`). The User Name and Password are set to the default administrator user (name="admin", password="YAWL").

As a user convenience, a button called **Test Connection** is supplied, which will attempt to connect to a running engine with the detail supplied before the user commits to using those details for any further editor/engine interactions.

Connecting to the Resource Service

In order to use the organizational model for assigning tasks to Participants or Roles, the connection between the editor and the Resource Service needs to be established.

To set the Resource Service connection details:

Click on the **Tools** menu and choose the **Resource Service Connection Settings…** item**.**

From the resulting dialog (Fig. 40), accept the default values or enter the following engine details:

- YAWL Engine URI
- Administrator's User Name
- Administrator's Password.

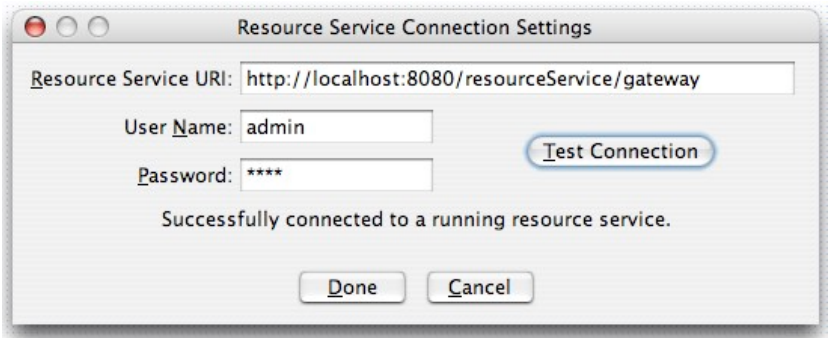


Fig. 40 Specifying the Resource Service connection

The Resource Engine value is set by default to a locally installed Resource Service (`http://localhost:8080/resourceService/gateway`). The User Name and Passwords are set to the default administrator user (name="admin", password="YAWL").

Connecting a Decomposition to a registered YAWL Service

You can use task decompositions within your workflow to make a connection to custom YAWL services that have been registered with a running engine.

For example, a decomposition may be set up to place an order with an external company. Upon execution of any task using this decomposition, data could be transmitted via a Web Service invocation to this company.

To have a decomposition invoke custom YAWL service, do the following:

3. Right-click on a task, the choose **Task Decomposition Detail…**. An Update Task Decomposition dialog box will appear (see the “YAWL Registered Service Detail” box of Fig. 29).

4. Select the required Web Service from the dropdown list of those available in the Engine.

5. Click **Done** to finish.

If the Editor is connected to a valid running YAWL Engine instance, the YAWL Service Dropdown Box will contain entries for all custom YAWL services the engine has registered. Otherwise, only the default (manual) “Worklist” YAWL service will be available.

When you select a YAWL Service, the editor will query this running engine for the input and output variables required, and populate the decomposition variables of the selected task with those variables. Base custom services that are supplied with the engine include one for RPC-Style Web Service Invocation, and one for making and receiving mobile phone SMS messages, and one for adding flexibility and exception handling support to YAWL processes.

Validating, Exporting and Importing

At any stage you can validate and/or export your specification to a YAWL Engine executable XML file.

To validate your specification:

1. Click on Validate this Specification button, , on the Menu Toolbar or click **Specification** on the Menu and choose **Validate Specification…**.

2. A table listing problems will appear with details of any inconsistencies that would stop the specification from running in the YAWL Engine in the specification problem panel (Fig. 41). Fig. 42 shows an example invalid specification.

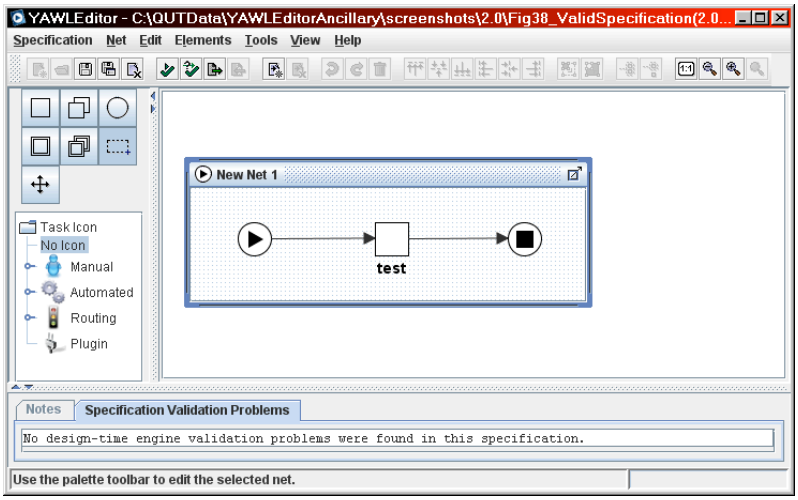


Fig. 41 A valid specification

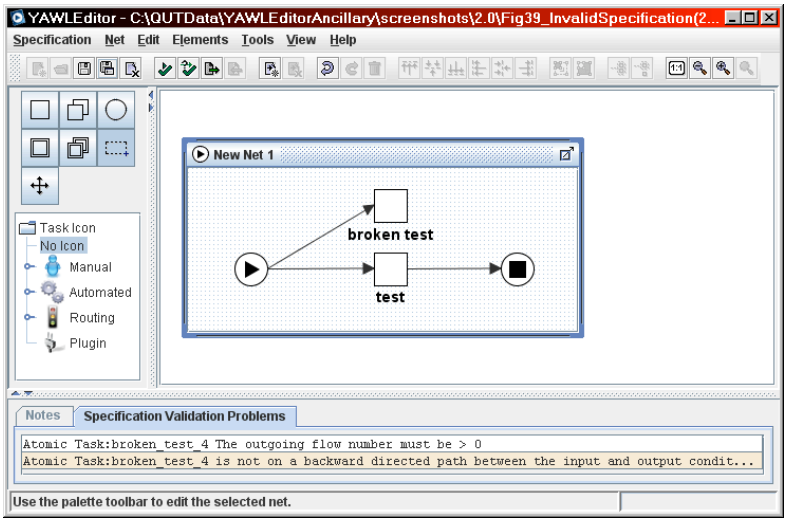


Fig. 42 An invalid specification

To export your specification to an XML file:

1. Click on the “Export this specification to the YAWL Engine file format” button, , on the Menu Toolbar or click **Specification** on the Menu and choose **Export to YAWL Engine File…**. The ‘Configure Export Settings’ dialog appears (Figure 43).

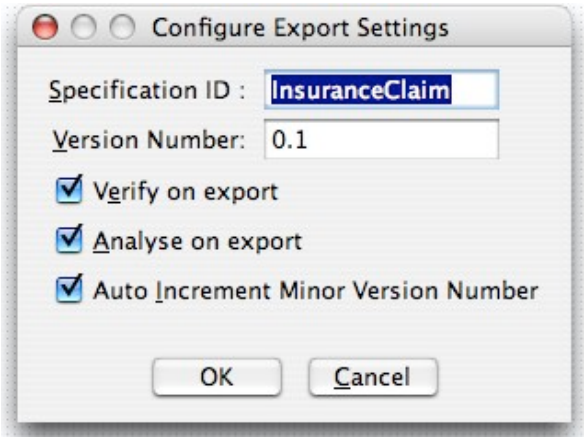


Fig. 43 Configure Export Settings dialog

If this is the first time the specification has been exported, and a ‘Specification ID’ has not yet been entered via the ‘Update Specification Settings’ dialog, a ‘Specification ID’ field will appear in the dialog (all Specifications must be given a valid ID). The ID can be a combination of alphanumeric characters and the underscore, but cannot start with a numeric character. Once set, this field will not reappear in subsequent exports of the specification.

The ‘Version Number’ field provides a version number for the specification. Version numbering allows different versions of the ‘same’ specification to co-exist in the Engine. YAWL specification version numbers consist of a major part (to the left of the point) representing a major revision, and a minor part (to the right of the point) representing a minor revision. Version numbers begin at 0.1; you may edit the version number directly to a *higher* version number at any time. If the ‘Auto Increment Minor Version Number’ checkbox is checked, the minor revision will increase by one each time the specification is exported.

The ‘Verify on export’ and ‘Analyse on export’ checkboxes, if checked, will verify and and/or analyse the specification before it is exported, providing a final check of the specification.

Once the dialog is completed, click **OK** – a file save dialog will appear asking you where to save the file.

2. Select a location and choose **Save**.

   This saved specification file can now be loaded into a running YAWL Engine and executed.

To import a specification from an XML file:

1. Click on the “Import YAWL Engine file” button, , on the Menu Toolbar or click **Specification** on the Menu and choose **Import from YAWL Engine File…**.

   A window will appear asking you specify the engine XML file.

2. Select a location and choose **Open**.

   This engine XML file is then loaded into the editor. As there is no layout and graphing detail on the engine XML file, the editor will apply basic layout to the imported specification.

Specification Analysis

Verification of specifications for the engine only determines whether the engine will be able to load the specification. In contrast, the analysis tool can be used to test for deeper issues in the specification.

The analysis toolbar button, , and matching **Analyse Specification** menu item under the **Specification** menu allows workflow designers to analyse their specifications. A number of potential problems with the workflow can be automatically spotted with analysis. Examples include spotting potential deadlock situations, unnecessary cancellations set members, and unnecessary or-join decorators (at run-time, or-joins require significant processing effort, and should be removed if they are not actually needed).

A configuration dialog for the specification analysis is available under the **Tools** menu, by the name of **Configure Specification Analysis** (see Figure 44).

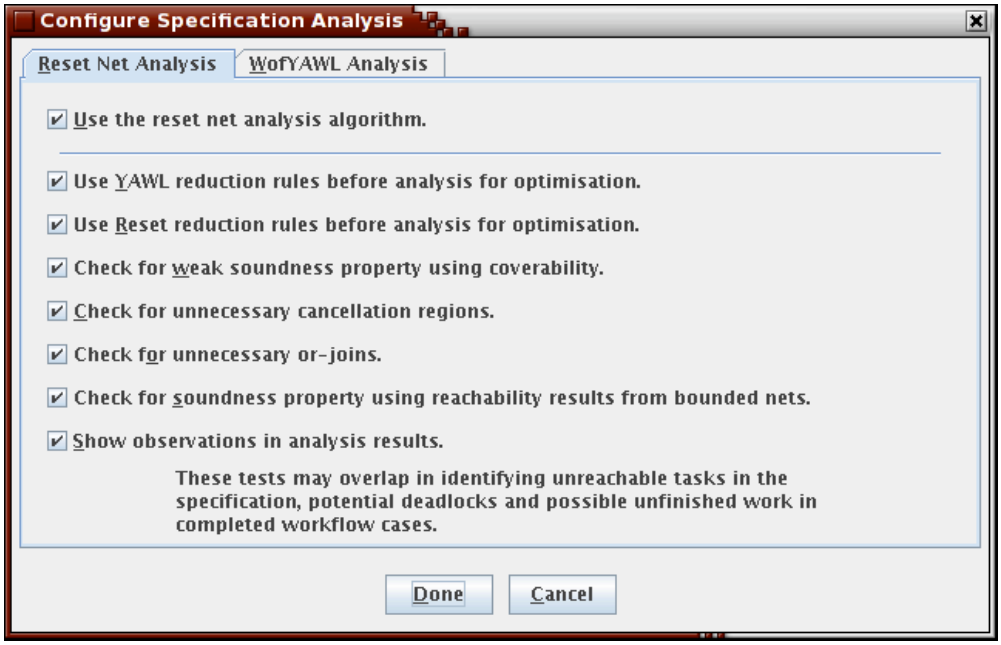

Configure Specification Analysis
Reset Net Analysis
WofYAWL Analysis
Use the reset net analysis algorithm.
Use YAWL reduction rules before analysis for optimisation.
Use Reset reduction rules before analysis for optimisation.
Check for weak soundness property using coverability.
Check for unnecessary cancellation regions.
Check for unnecessary or-joins.
Check for soundness property using reachability results from bounded nets.
Show observations in analysis results.
These tests may overlap in identifying unreachable tasks in the specification, potential deadlocks and possible unfinished work in completed workflow cases.
Done
Cancel

Fig. 44 Specification analysis configuration

If the optional YAWL specification analysis utility **wofyawl.exe**, written by Eric Verbeek is supplied in the same directory as the editor, an extra tab entitled **WofYAWL Analysis** will be enabled in this dialog, allowing more analysis options than those supplied by default. The utility must be compiled for specific architectures, and its data format has changed over time. This version of the editor needs version 0.4 of the utility, and expects it to be called **wofyawl0.4.exe**, regardless of the platform on which it runs. Choose the wofyawl implementation right for your environment from Sourceforge, and then rename as above it to get analysis working.

Automated task

Any non-composite task in YAWL can have its decomposition defined as *manual* or *automated*. A task with a manual decomposition is a task that is intended to be executed by a human resource, e.g. a participant in the organizational model. An task with an automated decomposition is a task that is not offered to any resource but is executed by the system. This type of task can be used to manipulate the content of net variables, from simple data assignments to complex reports generation. Alternately it may be associated with a *codelet* – a discrete piece of code that is executed, optionally using the input parameters of the task, and assigning any results to the chosen output parameters for the task.

Both task types are handled by the Resource Service, but the behaviour of an automated task differs as follows:

- on enablement, it is automatically checked out of the engine (thus having priority over manual tasks in a deferred choice) and its input parameters are parsed;
- if a codelet has been specified, it will be executed it using the task's parameters as required; then
- it is automatically checked in and its output parameters are parsed back to the corresponding net variables.

A task can be set as automated by right-clicking on it and selecting the **Task Decomposition Detail** dialog, then ticking the **Automated** checkbox in the **External Interaction** section (see Figure 29). When the Automated checkbox is checked, the option **Manage Resourcing** in the task's right click menu will be disabled (since human resources are not required for automated tasks).

Data manipulation can be achieved by using the task's variables and the task's Input and Output Parameters. Input Parameters are used to copy the content of a net variable onto a task variable. As Output Parameters, it is possible with an automated task to copy either from a task variable or from an XQuery expression onto a net variable. Fig. 45 shows an example of copying from expression, where the literal `true` is copied onto the net variable `PO_timedout` after choosing the from expression radio button. This choice is only available for automated tasks. Also, any task variables used in the XQuery expression itself must be declared as Input & Output types.

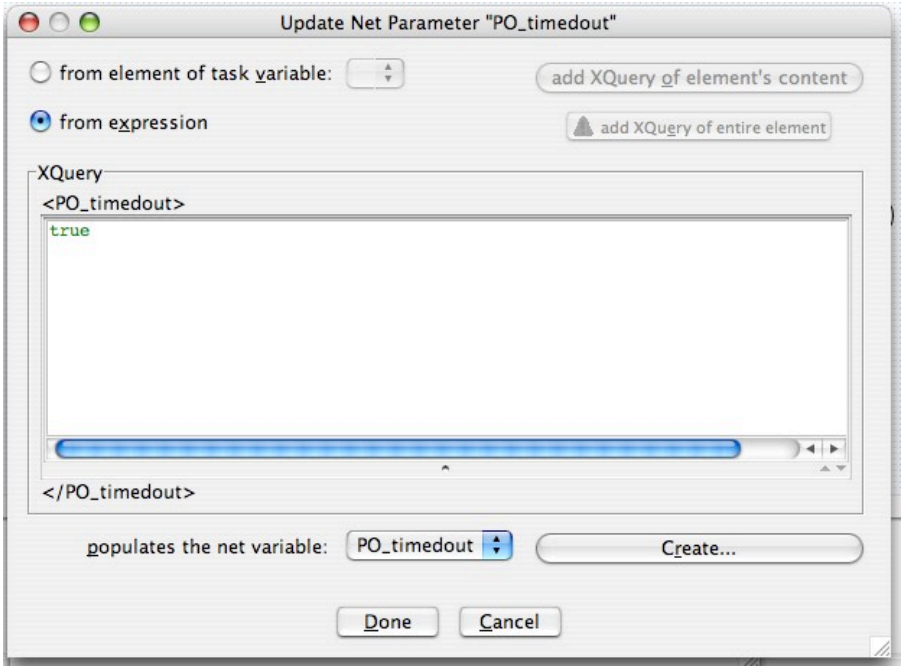


Fig. 45 An output parameter set to copy from expression for an automated task

Codelets

Essentially, a codelet is a discrete java class, managed by the resource service, that may be enacted by an automated task at runtime. When an automated task is enabled during process execution, and it has a codelet associated with it, the input parameters of the task are passed to the codelet, it is executed, and any results are passed back to the task via its output parameters.

There are currently three codelets available in the Editor:

- ExampleCodelet: designed to demonstrate the usage of codelets;
- XQueryEvaluator: accepts an XQuery as an input parameter, evaluates it (using other input parameters as required by the XQuery), and produces the result to an output parameter.
- ShellExecution: accepts an input parameter containing a command line of an external program, and runs it, waiting for it to complete and returning the result (if any) via an output parameter.

To associate a codelet with a task decomposition, first mark the decomposition as automated by ticking the **Automated** checkbox in the Task Decomposition dialog (see Figure 29). When checked, the **Set Codelet** button is enabled; clicking that button opens the *Set Codelet for Automated Decomposition* dialog appears, listing the available codelets (Figure 46).

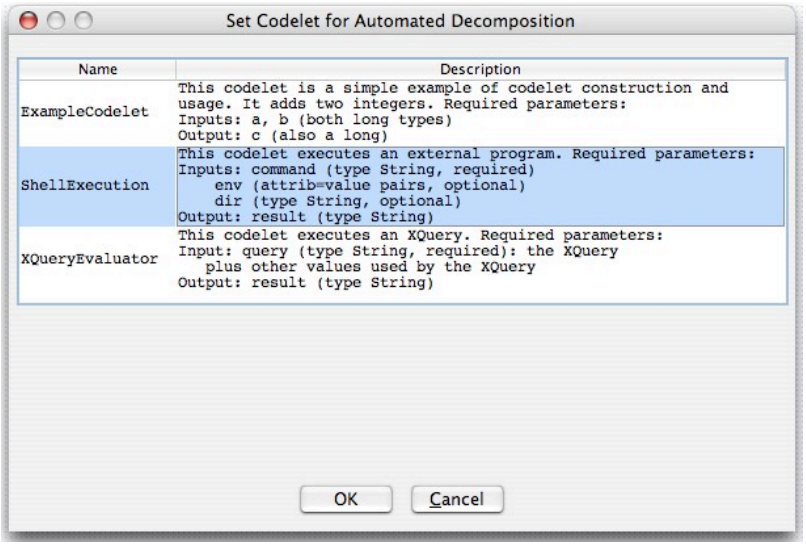

Set Codelet for Automated Decomposition
Name
Description
ExampleCodelet
This codelet is a simple example of codelet construction and usage. It adds two integers. Required parameters:
Inputs: a, b (both long types)
Output: c (also a long)
ShellExecution
This codelet executes an external program. Required parameters:
Inputs: command (type String, required)
env (attrib=value pairs, optional)
dir (type String, optional)
Output: result (type String)
XQueryEvaluator
This codelet executes an XQuery. Required parameters:
Input: query (type String, required): the XQuery
plus other values used by the XQuery
Output: result (type String)
OK
Cancel

Fig. 46 The Set Codelet dialog

The dialog lists the available codelets, together with a description of what each one does and the parameters required to successfully execute it. For example, if ExampleCodelet is chosen, the automated task requires 3 parameters to be created: input parameters ‘a’ and ‘b’, and output parameter ‘c’ (all of type ‘long’). These parameters must be added to the task in the normal manner – at runtime, if the required parameters are not present, the codelet will be unable to successfully complete (the task will still complete successfully, however).

The codelet repository has been designed as ‘pluggable’, so that designers and developers can easily add new codelets to perform various operation, which will immediately be available to process designers via the dialog above.

Manual task - Resource Management

Once a connection with the Resource Service has been established, any non-composite, manual task (a task is manual by default, and only becomes automated when explicitly checked as automated in the Task Decomposition Dialog) can allocated resources in a number of steps, by right-clicking on it and selecting **Manage Resourcing**. This will launch the resource manager wizard. Fig. 47 shows the dialog window for Step 1 of the wizard.

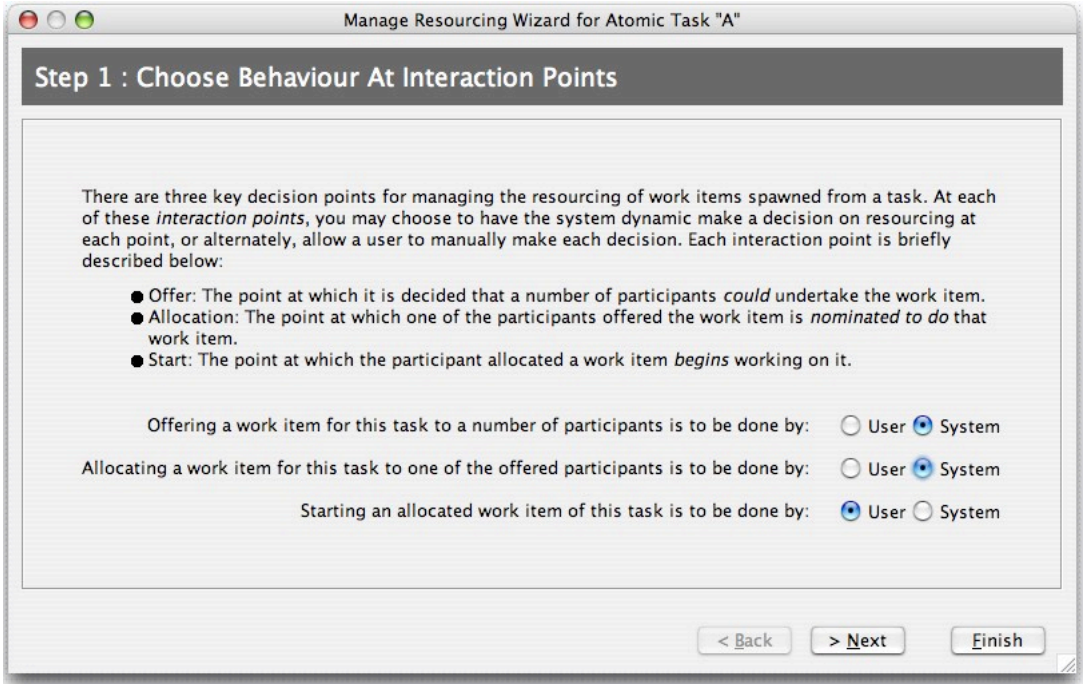

Manage Resourcing Wizard for Atomic Task "A"
Step 1 : Choose Behaviour At Interaction Points
There are three key decision points for managing the resourcing of work items spawned from a task. At each of these interaction points, you may choose to have the system dynamic make a decision on resourcing at each point, or alternately, allow a user to manually make each decision. Each interaction point is briefly described below:
Offer: The point at which it is decided that a number of participants could undertake the work item.
Allocation: The point at which one of the participants offered the work item is nominated to do that work item.
Start: The point at which the participant allocated a work item begins working on it.
Offering a work item for this task to a number of participants is to be done by: User System
Allocating a work item for this task to one of the offered participants is to be done by: User System
Starting an allocated work item of this task is to be done by: User System
< Back
> Next
Finish

Fig. 47: Step 1 of the resource management wizard

Step 1

In Step 1 we can specify the interaction strategy for work items of the selected task. There are three interaction points: offer, allocation, and start, each of which may be either User or System processed.

If we choose that offered work items are dealt with by the User and not by the System, at run time an Administrator will have to decide whom to offer the work item to. Otherwise, in Steps 2 and 3, we can specify to whom work items have to be offered by the System.

If we choose that allocated work items are dealt with by the User and not by the System, any participant who has been offered the work item, can choose whether to commit to future execution of this work item (i.e. to allocate the work item to themselves). Otherwise, in Step 4 we can specify how work items have to be offered by the System.

Step 2

In Step 2, shown in Fig. 48, we can select an initial set of resources that will be offered work items of the selected task at runtime by the System. The set of resources can be determined by a combination of Participants and/or Roles. These can be picked from the respective lists and/or from net-level variables of type string, which at runtime will hold the userid of a participant or the name of a role (deferred allocation). Note that all net-level variables will be listed in the *Net Parameters* table, and will initially all be set as referring to 'Data'. For each of those that will contain resourcing information at runtime, the *Refers To* value must be changed to either Participant or Role via the drop down list on each row, depending on the type of resource the value will refer to.

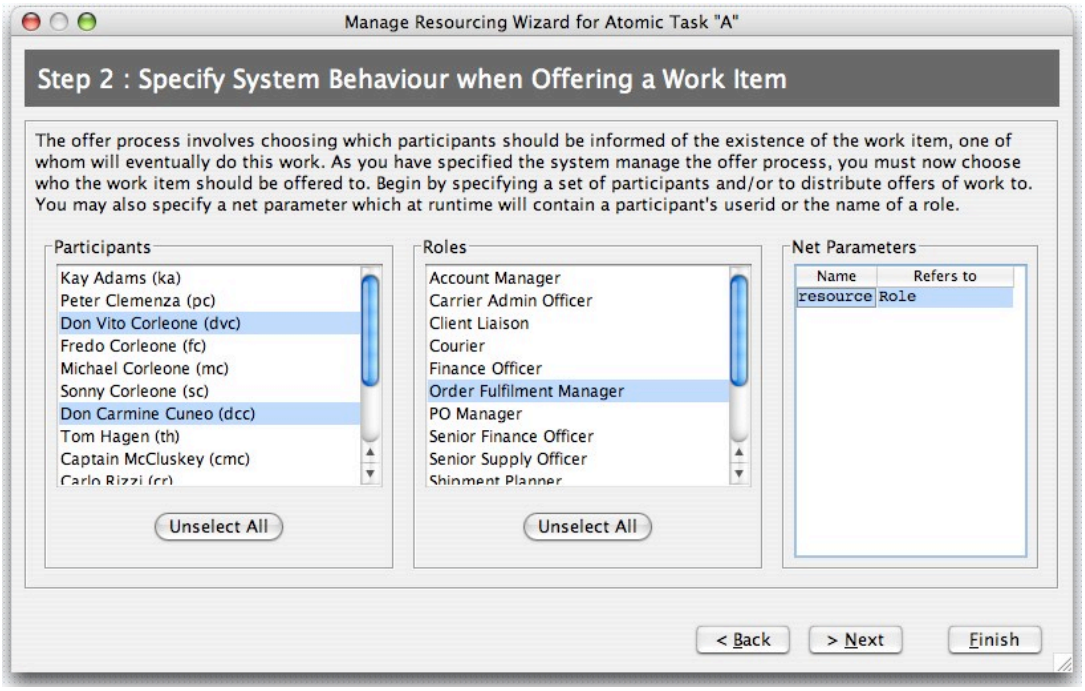

Manage Resourcing Wizard for Atomic Task "A"
Step 2 : Specify System Behaviour when Offering a Work Item
The offer process involves choosing which participants should be informed of the existence of the work item, one of whom will eventually do this work. As you have specified the system manage the offer process, you must now choose who the work item should be offered to. Begin by specifying a set of participants and/or to distribute offers of work to. You may also specify a net parameter which at runtime will contain a participant's userid or the name of a role.
Participants
Kay Adams (ka)
Peter Clemenza (pc)
Don Vito Corleone (dvc)
Fredo Corleone (fc)
Michael Corleone (mc)
Sonny Corleone (sc)
Don Carmine Cuneo (dcc)
Tom Hagen (th)
Captain McCluskey (cmc)
Unselect All
Roles
Account Manager
Carrier Admin Officer
Client Liaison
Courier
Finance Officer
Order Fulfilment Manager
PO Manager
Senior Finance Officer
Senior Supply Officer
Unselect All
Net Parameters
Name
Refers to
resource
Role
< Back
> Next
Finish

Fig. 48: Step 2 of the resource management wizard

Step 3

In Step 3, shown in Fig. 49, we can filter the set of participants that have been described in Step 2. Filtering can be done over capabilities and/over organizational data, which consists of organizational groups and positions. Moreover, from this dialog it is possible to:

- allow the work items of the selected task to be offered only to participants who are members of the set specified in Step 2 and have previously completed work items of another task (as specified) in the current process instance;
- do *not* allow the work items of the selected task to be offered to participants who are members of the set specified in Step 2 and have previously completed work items of another task (as specified) in the current process instance;

- Allow all of the work items of the selected task to be *piled* to a particular user, which means that, if a user chooses to pile a work item of the task, that user will be automatically allocated the work item *and all future instances of workitems of the task for all future instances of the process containing the task*, until such time as piling is turned off for that task by the user or an administrator.

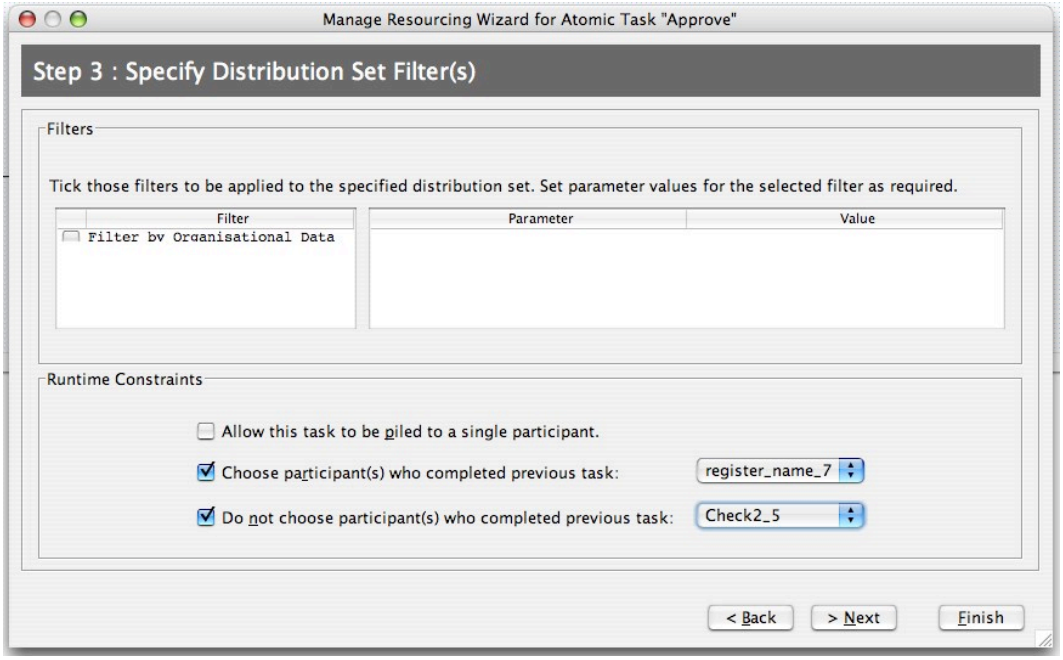

Manage Resourcing Wizard for Atomic Task "Approve"
Step 3 : Specify Distribution Set Filter(s)
Filters
Tick those filters to be applied to the specified distribution set. Set parameter values for the selected filter as required.
Filter
Parameter
Value
Filter by Organisational Data
Runtime Constraints
Allow this task to be piled to a single participant.
Choose participant(s) who completed previous task:
register_name_7
Do not choose participant(s) who completed previous task:
Check2_5
< Back
> Next
Finish

Fig. 49: Step 3 of the resource management wizard

Step 4

In Step 4, shown in Fig. 50, we can select the allocation mechanism that determines at run time which participant, among the members of the distribution set, will actually be allocated the work item. The currently available mechanisms are:

- Round Robin (by time): chooses the participant in the set who has not been allocated a workitem for the longest time;
- Round Robin (by frequency): chooses the participant in the set who has been allocated the least number of work items
- Random Choice: randomly chooses a participant from the set (the default); and
- Shortest Queue: chooses the participant in the set who has the least number of work items currently in their work queue.

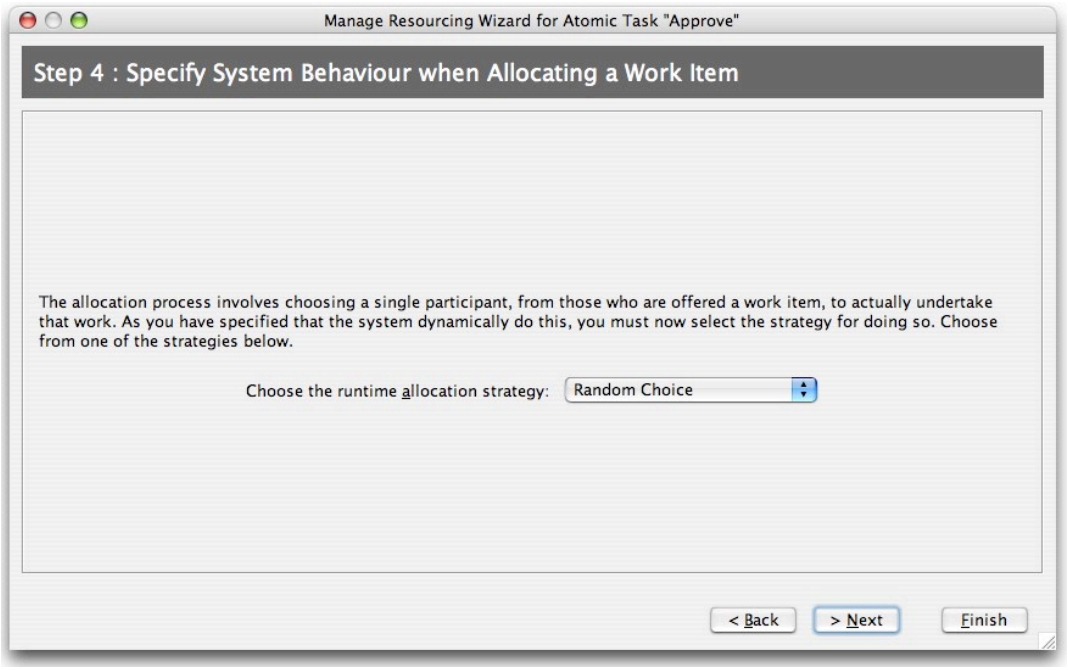

Manage Resourcing Wizard for Atomic Task "Approve"
Step 4 : Specify System Behaviour when Allocating a Work Item
The allocation process involves choosing a single participant, from those who are offered a work item, to actually undertake that work. As you have specified that the system dynamically do this, you must now select the strategy for doing so. Choose from one of the strategies below.
Choose the runtime allocation strategy:
Random Choice
< Back
> Next
Finish

Fig. 50: Step 4 of the resource management wizard

Step 5

In Step 5, shown in Fig. 51, we can specify participant-task privileges (privileges that apply only for this task). For example, we can specify whether participants are allowed to suspend the execution of work items of the selected task.

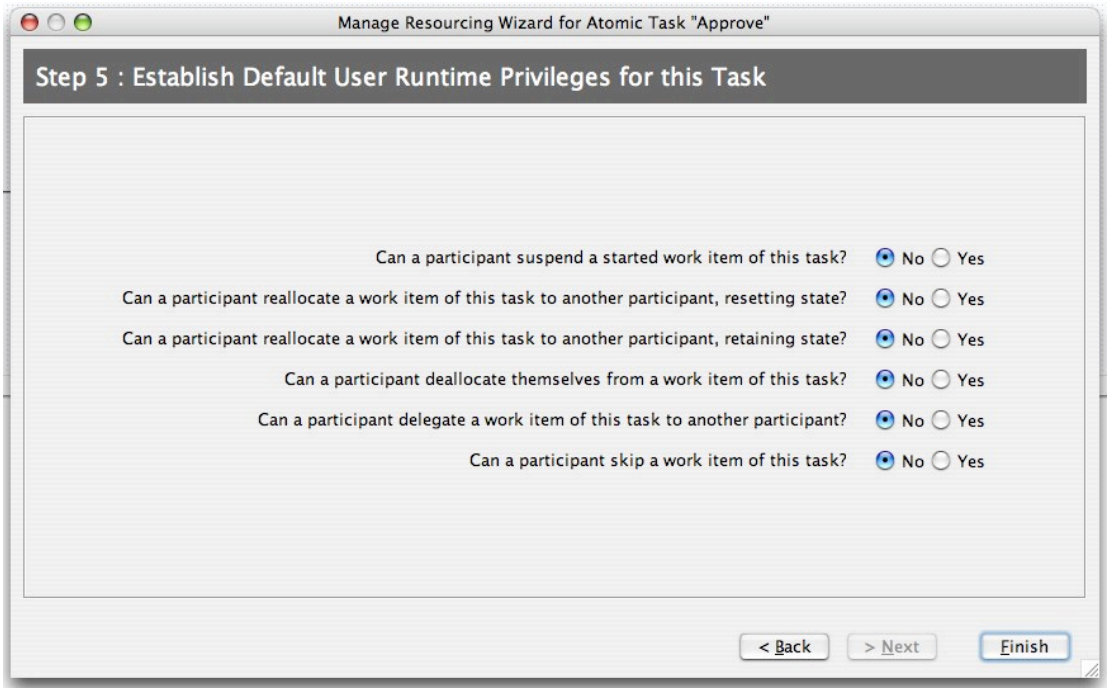

Manage Resourcing Wizard for Atomic Task "Approve"
Step 5 : Establish Default User Runtime Privileges for this Task
Can a participant suspend a started work item of this task? No Yes
Can a participant reallocate a work item of this task to another participant, resetting state? No Yes
Can a participant reallocate a work item of this task to another participant, retaining state? No Yes
Can a participant deallocate themselves from a work item of this task? No Yes
Can a participant delegate a work item of this task to another participant? No Yes
Can a participant skip a work item of this task? No Yes
< Back
> Next
Finish

Fig. 50: Step 5 of the resource management wizard

More details on engine run-time user allocation and authorisation can be found in the YAWL Engine manual and in the Resource Perspective Requirement Analysis document.

More details on populating the organisational database of the engine can be found in the YAWL administration tool manual.

Timeout Task

Any atomic task can be given a timeout behaviour by right-clicking on it and selecting the **Set Task Timer…**. The dialog in Fig. 52 will appear.

From this dialog it is possible to set an activation type and an expiry value for the timeout. The timeout can be activated (i.e. fired) upon work item enablement or upon work item starting. These have different meanings according to the type of task – manual vs. automated.

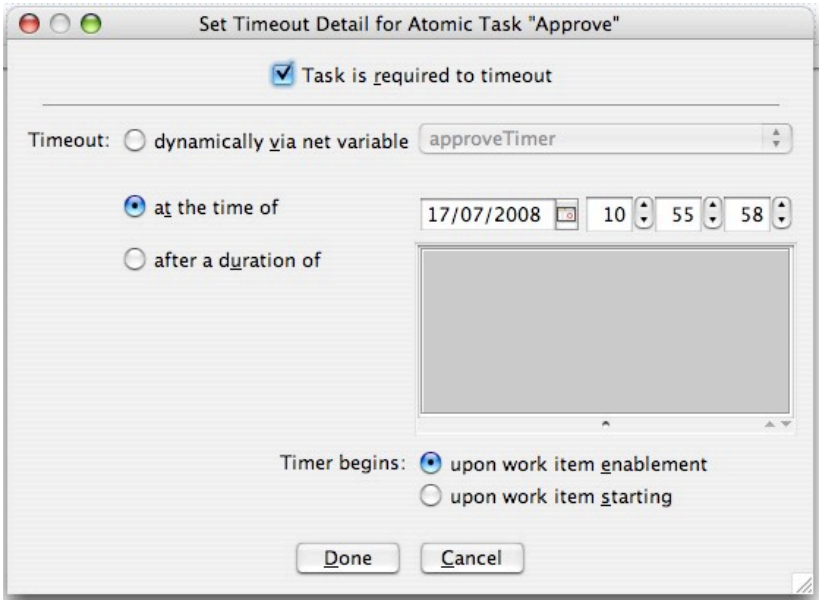


Fig. 51 The timeout dialog for an atomic task

Activation on enablement

- In the case of a manual task, as soon as the task is enabled, the timer begins and it remains live so long as the timeout does not expire. During this time frame, the task will follow the normal resource assignment policy. In other words, it will be offered and can be allocated and started. Once the timeout expires, the work item will complete no matter what the current status is (offered, allocated, started). The possible danger of this behaviour is that a work item might be timed out while being edited by a user.

- In the case of an automated task, the timeout works as a delay, i.e. the automatic execution of the work item created by an automated task is delayed by the value of the timeout. Once the timeout expires, the task is executed and then completed.

Activation on starting

- In case of a manual task, the timeout only starts once the task has started. Therefore, the task will be first offered, then allocated, and once it is started the timeout starts. Again, the timeout may expire while the task is being edited by a user.

- This option is does not apply for an automated task.

Expiry value

The expiry value of the timeout indicates for how long the timeout will live after being activated. Expiry values can be expressed in two ways:

- Using a specific date and time, which means the timeout will occur at the specified moment. To set a specific date and time, choose the 'at the time of' radio button on the Set Timeout Detail dialog (fig.52) and enter the required values. Care should be taken when setting a specific time value for timeouts – if it happens that the specified value is earlier than when the task is actually enabled or started, then the YAWL Engine will recognise that the timeout has already expired and immediately complete the workitem before it has a chance to appear on a worklist.

- Using a Duration value. A Duration is one of the XML Schema data types, and is used to represent a period of time. When a Duration type is used as a timeout value, the timeout will occur exactly when that period of time has passed since the workitem was enabled or started (depending on when it activates). To set a specific duration, choose the 'after a duration of' radio button on the Set Timeout Detail dialog (fig.52) and enter the required value. A Duration value is expressed in the following form:

  P*n*Y*n*M*n*DT*n*H*n*M*n*S

  All values start with P (for Period) followed by a non-negative number of years, months, days, then T (for time), followed by hours, minutes and seconds. The seconds value can have a decimal point and as many digits following the point as required (e.g. to specify milliseconds). Any zero value parts can be omitted. Valid examples: P1Y4M3DT23H55M1.5S, P2M3D, PT10S. Note that P2M may mean a different number of days depending on what month it is started in.

In addition to the methods described above, timer parameters may also be set at run-time via a declared variable of type YTimerType, so that values can be supplied and late-bound to a task's timer settings. To use this deferred approach:

1. Create a net-level parameter of type YTimerType.

2. Choose a task, right click on it and choose **Set Task Timer…**. to open the Set Timeout Detail dialog (Fig. 52), click the 'dynamically via net variable' option and select the net-level parameter created above.

3. Map the net-level parameter to another (previous) task in the flow, which will be used to capture the required values at runtime. Those values will be used to set the timer parameters on the task selected in Step 2. The values requested are:

    - **Trigger**: when should the timer start? There are two valid trigger values, *OnEnabled* and *OnExecuting*.

    - **Expiry**: when should the timer expire? This value can be either a date string (for example 12/12/08), which will be interpreted as a specific timeout, or as a Duration value, which will be interpreted as a period of time.

Extended Attributes

The editor offers a mechanism for defining extended attributed to be associated to task decompositions, net variables and task variables. The implementation and the semantics of these attributes is left to the developer.

Extended attributes can be defined in property files, which need to be placed in the folder:

```
<editor_install_path>/YAWLEditorPlugins/ExtendedAttributeProperties.
```

Attributes referring to net and task variables must be defined in a file named `VariableProperties` (no extension), while attributes referring to task decompositions must be defined in a file named `DecompositionProperties` (no extension). The files are read at the editor's next restart.

Attributes can be of type `string`, `boolean`, `enumeration` and `XQuery`. The following is an example of property file that defines the attributes `description`, `help`, `mode`, `refresh` and `skipSchemaValidation` (comments are indicated by a line starting with #).

```
#Decomposition Attributes
#Wed May 14 17:35:42 AET 2008
description=xquery
help=string
mode=enumeration{normal,final,pending}
refresh=xquery
skipSchemaValidation=boolean
```

From the editor, the user can visualize and edit the list of attributes by selecting the tab **Extended Attributes** from one of the following dialog windows:

- Update Task Decomposition,

- Update Net Variable,

- Update Task Variable.

Fig. 53 shows the rendered list of attributes for the above property file.

Note that attributes can only be specified through property files and not from the above files.

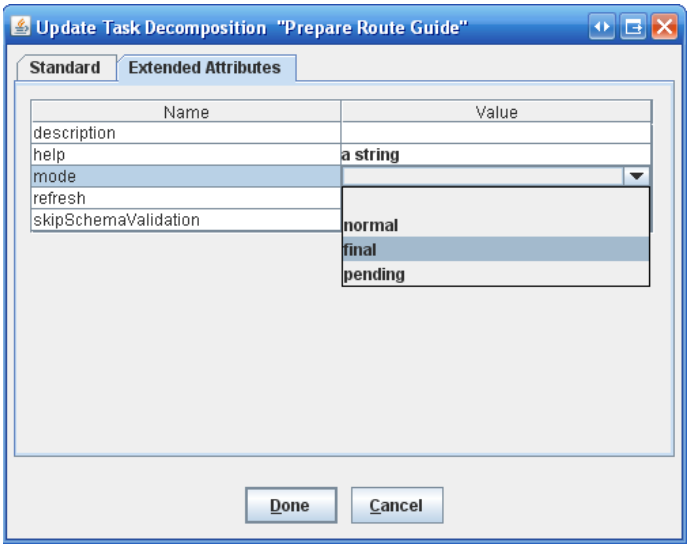


Fig. 52 The dialog for editing the extended attributes associated to task decompositions

Illustrative Examples

Multiple Instance Task Example

The example save file below, supplied with the editor source, gives you an example of how to process a simple multiple instance task, with its queries filled in (Fig 54).

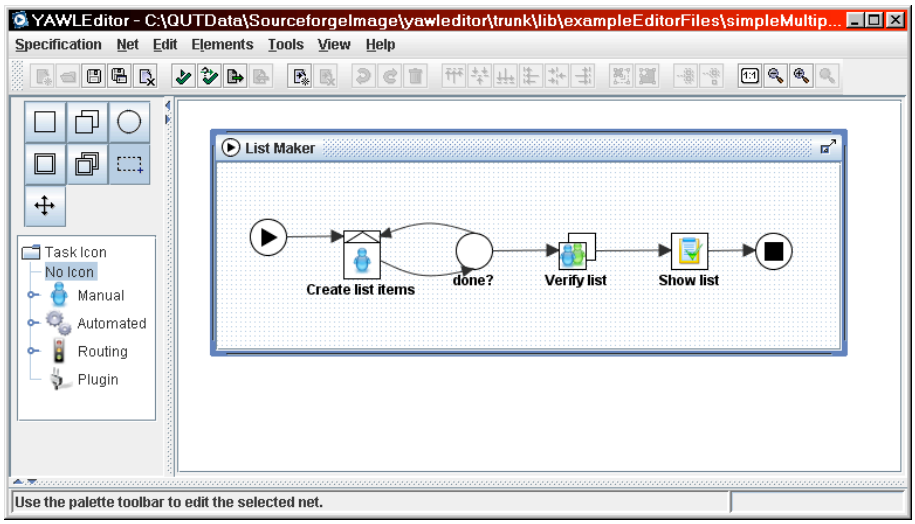


Fig. 53: A simple specification showcasing multiple instance tasks

Fig. shows the core multiple instance queries used to manipulate a XML Schema element defined for this specification. The element has a complex type of “Person-List” which defines a sequence of “<PersonName>someName</PersonName>” elements.

Timeout Example

This example shows a timeout set on the enablement of an automated task. The purpose of this task is to allow users a given time frame for the modification or confirmation of a purchase order. This situation is shown in the sub-net Ordering of Fig. 56, where after the creation and the approval of the purchase order, the process flow leads to a deferred choice among tasks Modify Purchase Order, Confirm Purchase Order and Order Timeout.

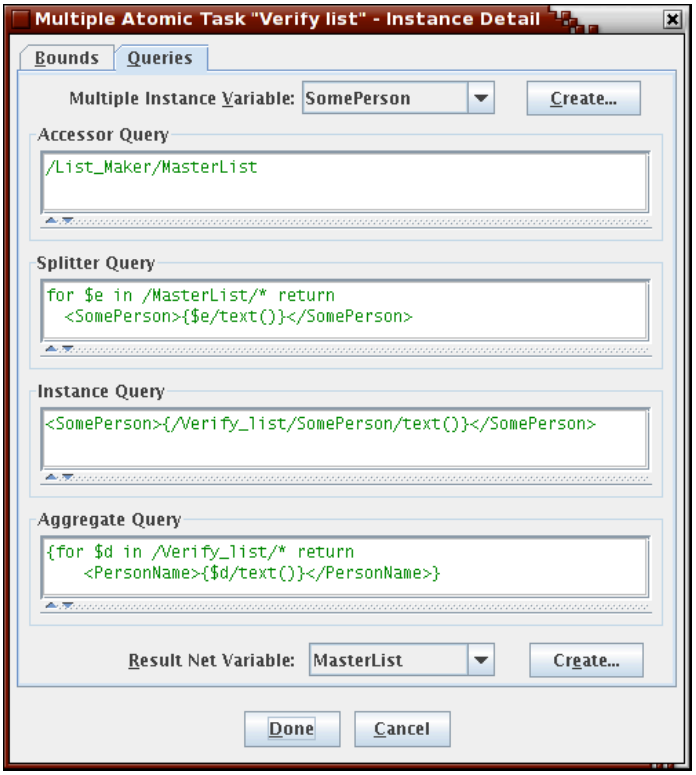


Fig. 55 Example of multiple instance queries

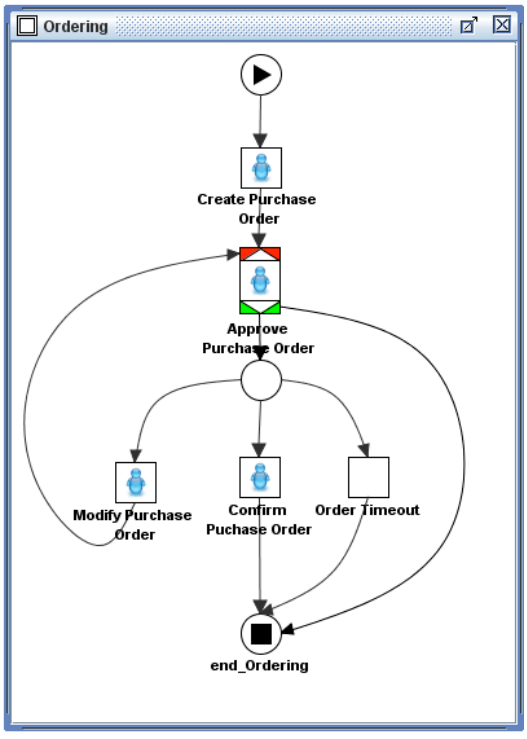


Fig. 56 Example of timeout on enablement for an automated task

Known Issues

- Users cannot type double quote characters in XQuery and XPath expressions (this is a deliberate restriction to stop oddities in engine export manifesting). Workflow designers are to use single quote characters when manipulating string literals in XQuery/XPath expressions.
- Editor Import functionality does not currently import resource perspective or timer detail from an engine XML specification file.
- The menu item "Print Specification" freezes if not used for some time.

Troubleshooting

The YAWL Editor is not responding to my mouse clicks and it is beeping every time I click the mouse.

Check to see if you have any YAWL Editor dialog windows open. These windows don't appear in the Windows Taskbar. To check if you have any open dialog windows, hold down the ALT key and press the TAB key. With the ALT key held down, press TAB until you reach the dialog window and let go of the ALT key. Close the dialog window and you should be able use the YAWL Editor again.

Dialog windows appear as the Java icon (blue coffee cup).

I can't connect two elements with a flow. Why?

If the editor does not allow you to connect one element to another, it is steering you away from building an invalid net. Typical examples include:

Trying to connect a second flow to an undecorated task.

Trying to point an incoming flow to a split decorator (or an outgoing flow from a join decorator).

Se the troubleshooting entry "How do I find out more about elements and principles....." for more detail.

When I validate my Net, I get the following validation message, 'The net (. . .) may complete without any generated work. Should all atomic tasks in the net be unlabelled?'

This message appears if you have not labelled any of your tasks. To label a task (and thus ensure that the engine will get a user to handle the task at this point in the workflow), right click on the task and choose Select Task Decomposition…. Choose Create… and in the following window, "Update Task Decomposition", and enter the name of your task under Decomposition Label, then click Done.

To label a Condition, right-click on a task and choose Set Label…, then type in a name for the label and click Done.

My specification won't validate without any errors and I don't know what to do.

First, check the logic of your specification and exercise every branch of your workflow for correctness.

If you are sure of your workflow, consult the YAWL website for the proper use of the YAWL elements:

http://yawlfoundation.org.

How do I find out more about the elements and principles of the YAWL workflow specifications?

For more information about the mechanics of the YAWL workflow elements, please consult the YAWL website:

http://yawlfoundation.org .

Copyright Notice

Copyright to this software and its source code is granted under the GNU Lesser General Public Licence (v2.1).

For detail on the permissions this licence grants you, please refer to

http://www.gnu.org/copyleft/lesser.html.

This editor makes use of JGraph 5.12.1.1. This version of the JGraph library is covered under an LGPL-style licence and is available separately from:

http://www.jgraph.com/.

This editor also makes use of code from the YAWL Engine and its support libraries. The engine and its support libraries are covered under their own individual copyright licences, detail of which may be found at the YAWL website:

http://www.yawlfoundation.org.

In accordance with this editor's copyright licence, the source code may be obtained from the YAWL SourceForge website:

http://sourceforge.net/projects/yawl/.

Acknowledgements

This documentation includes feedback from Marlon Dumas, Lachlan Aldred and Arthur ter Hofstede.